OPTIMIZATION WITH PREDICTIONS AND/OR ONLINE

CHRISTIAN COESTER (OXFORD)

Traditional Algorithms

Worst-case guarantees

Pessimistic?

Machine learned predictions

Often very powerful

No guarantee

Traditional Algorithms

Worst-case guarantees

Pessimistic?

Machine learned predictions

Often very powerful

No guarantee

Real life $\neq$ worst case, often predictable

(e.g., solve similar instances repeatedly)

Traditional Algorithms

Worst-case guarantees

Pessimistic?

Machine learned predictions

Often very powerful

No guarantee

Real life $\neq$ worst case, often predictable
(e.g., solve similar instances repeatedly)

Algorithms with predictions

Goal: Good predictions $\Longrightarrow$ much better performance

Bad predictions $\Longrightarrow$ same worst-case guarantee

Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]

2	5	7	10	16	23	28	36	37	42	47	58	60	67	73	80	83

Does 67 appear in array?

Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]

2	5	7	10	16	23	28	36	37	42	47	58	60	67	73	80	83

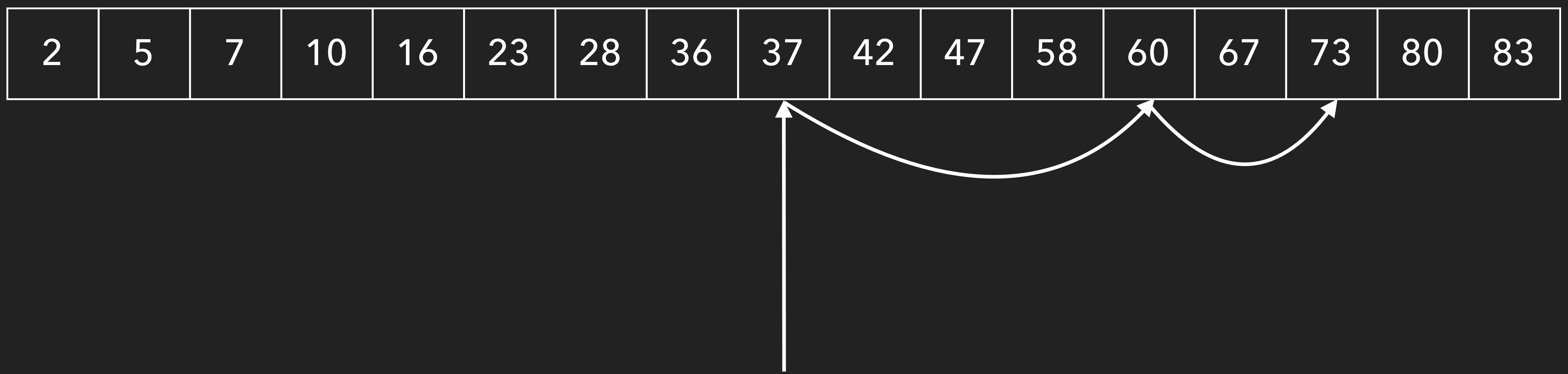
Does 67 appear in array?

Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]

2	5	7	10	16	23	28	36	37	42	47	58	60	67	73	80	83

Does 67 appear in array?

Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]



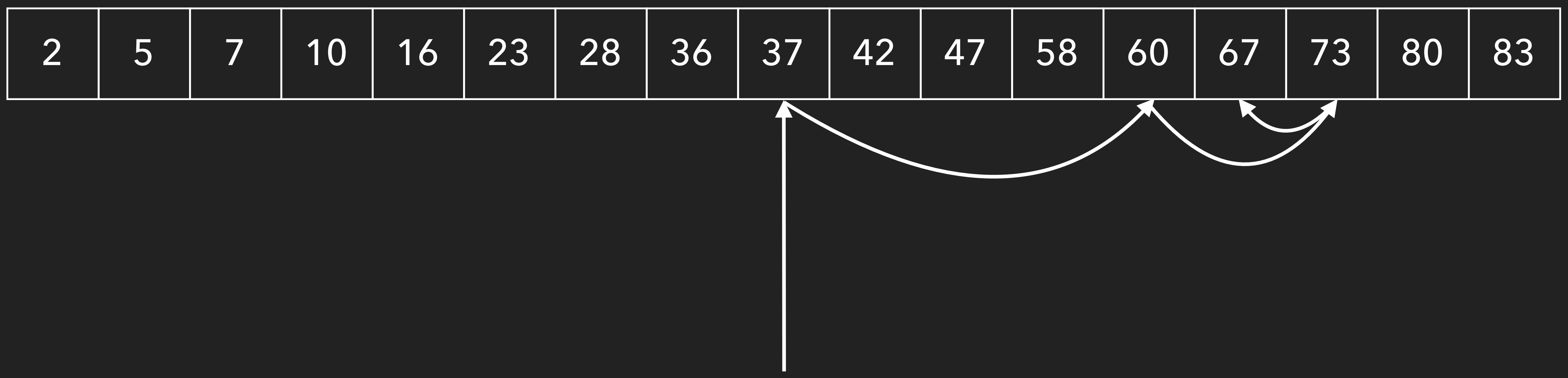
Does 67 appear in array?

Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]

2	5	7	10	16	23	28	36	37	42	47	58	60	67	73	80	83

Does 67 appear in array?

Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]



Does 67 appear in array?

- Binary search: Time $O(\log n)$

Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]

											$\hat{p}$		p			
2	5	7	10	16	23	28	36	37	42	47	58	60	67	73	80	83

Does 67 appear in array?

- Binary search: Time $O(\log n)$
- Given prediction $\hat{p}$ of position p

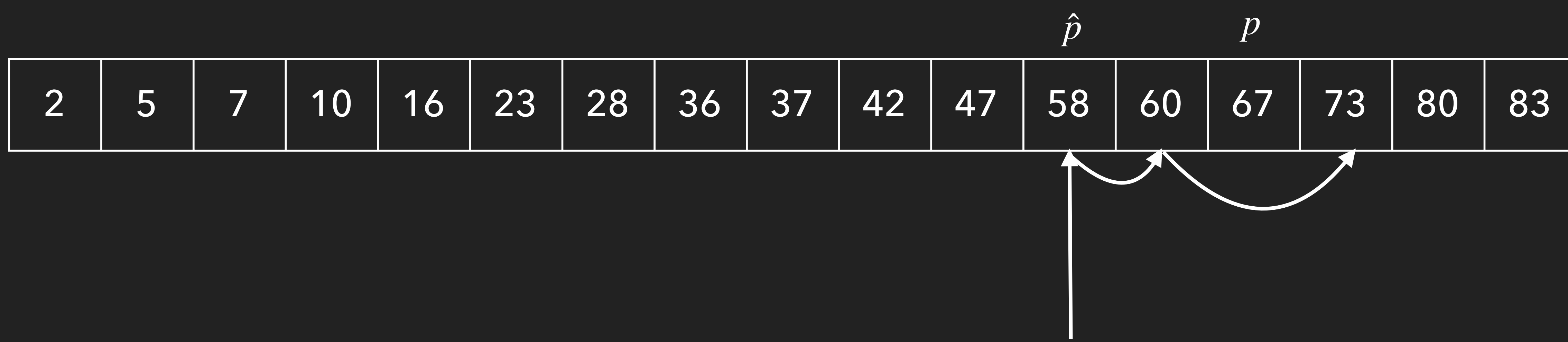
Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]

											$\hat{p}$		p			
2	5	7	10	16	23	28	36	37	42	47	58	60	67	73	80	83

Does 67 appear in array?

- Binary search: Time $O(\log n)$
- Given prediction $\hat{p}$ of position p

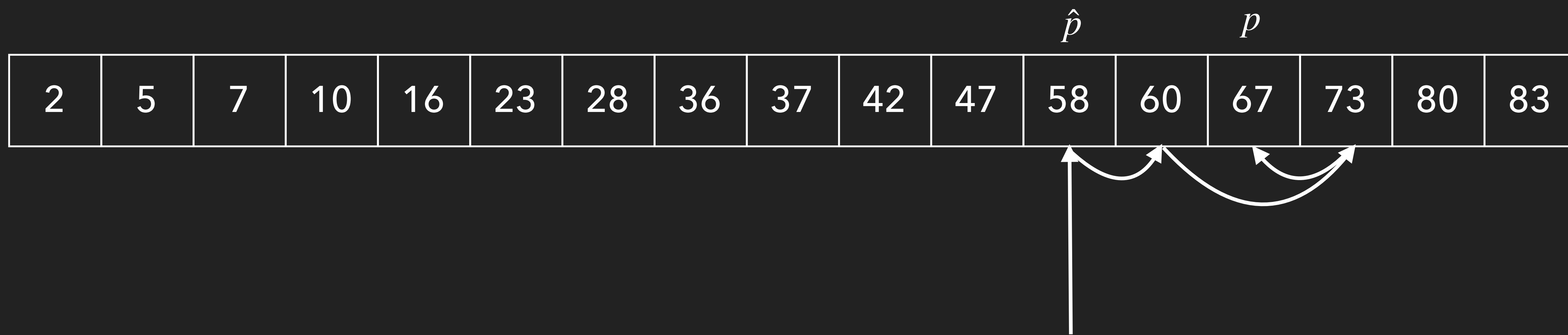
Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]



Does 67 appear in array?

- Binary search: Time $O(\log n)$
- Given prediction $\hat{p}$ of position p

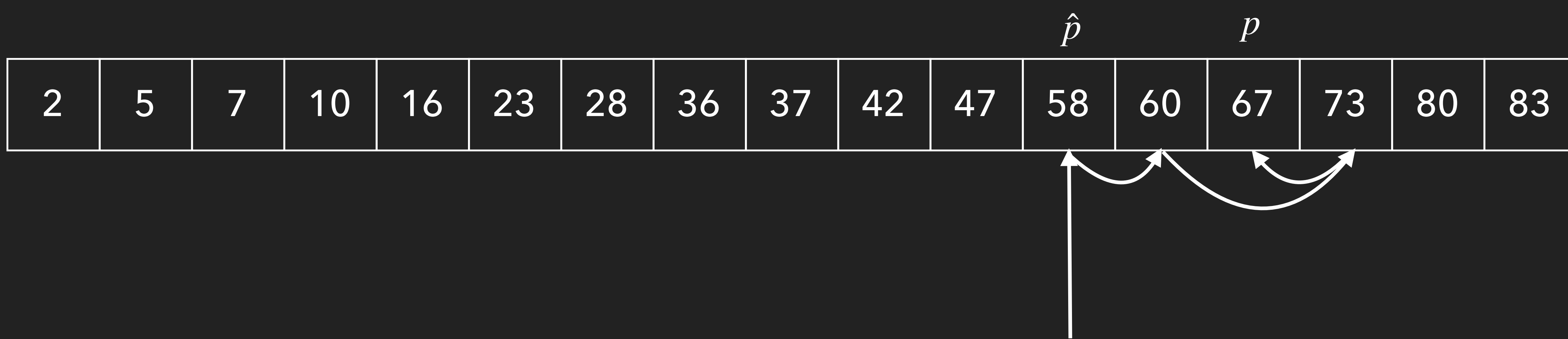
Example: Binary Search [Lykouris, Vassilvitskii 18] [Kraska et al. 18]



Does 67 appear in array?

- Binary search: Time $O(\log n)$
- Given prediction $\hat{p}$ of position p

Example: Binary Search [Lykouris,Vassilvitskii 18] [Kraska et al. 18]



Does 67 appear in array?

- Binary search: Time $O(\log n)$
- Given prediction $\hat{p}$ of position p
  - Time $O(\log \eta)$, where $\eta = |\hat{p} - p|$

Algorithms with Predictions (aka Learning-Augmented Algorithms)

Growing rapidly in last 5 years

- Caching
- Scheduling/Load Balancing
- Rent or buy problems
- Metrical Task Systems
- Matching
- Data Structures
- ...

Improve competitive, approx. ratio, running time, space, ...

Sorting with Predictions

JOINT WORK WITH XINGJIAN BAI (OXFORD)

Sorting with Predictions [Bai,Coester 23]

Task: Sort $a_1, a_2, \ldots, a_n$ wrt. $<$

Setting 1: Receive prediction of positions in sorted list

Setting 2: Access to quick-and-dirty comparisons

Sorting with Positional Predictions

Input: $a_1, a_2, \ldots, a_n$

prediction $\hat{p}(i)$ of a_i's position in sorted list

Sorting with Positional Predictions

Input: $a_1, a_2, \ldots, a_n$

prediction $\hat{p}(i)$ of a_i's position in sorted list

Similar: Adaptive Sorting

We consider element-wise error $\rightsquigarrow$ fine-grained guarantees

Sorting with Positional Predictions

Input: $a_1, a_2, \ldots, a_n$

prediction $\hat{p}(i)$ of a_i's position in sorted list

Similar: Adaptive Sorting

We consider element-wise error $\rightsquigarrow$ fine-grained guarantees

Notation: $p(i) =$ true position of a_i in sorted list

$$\eta_i = |\hat{p}(i) - p(i)|$$

Sorting with Positional Predictions

Input: $a_1, a_2, \ldots, a_n$

prediction $\hat{p}(i)$ of a_i's position in sorted list

Similar: Adaptive Sorting

We consider element-wise error $\rightsquigarrow$ fine-grained guarantees

Notation: $p(i) =$ true position of a_i in sorted list

$$\eta_i = |\hat{p}(i) - p(i)|$$

Theorem: $\exists$ algorithm that sorts in time $O\left(\sum_{i=1}^{n} \log(\eta_i + 2)\right)$

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

First algorithm:

1. Bucket sort according to $\hat{p}$
2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	82		208	281		364		510			621	711		813		914
	67					398		491			649					894
	90					385					625					

First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	82		208	281		364		510			621	711		813		914
	67					398		491			649					894
						385					625					

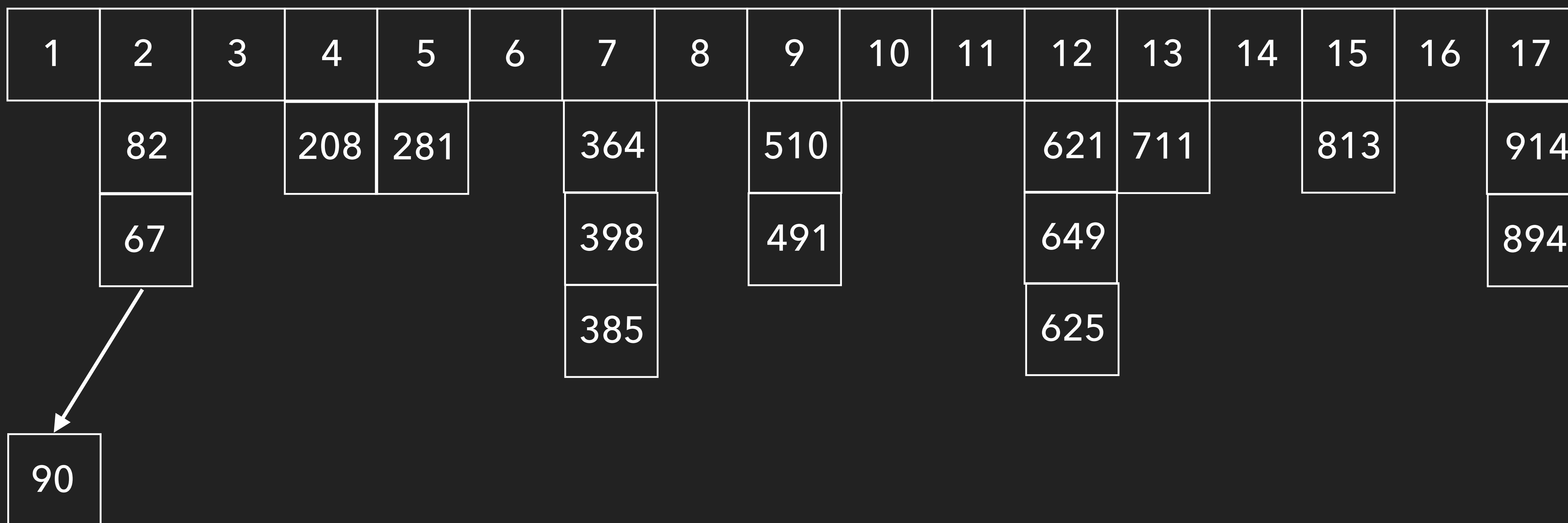
90

First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



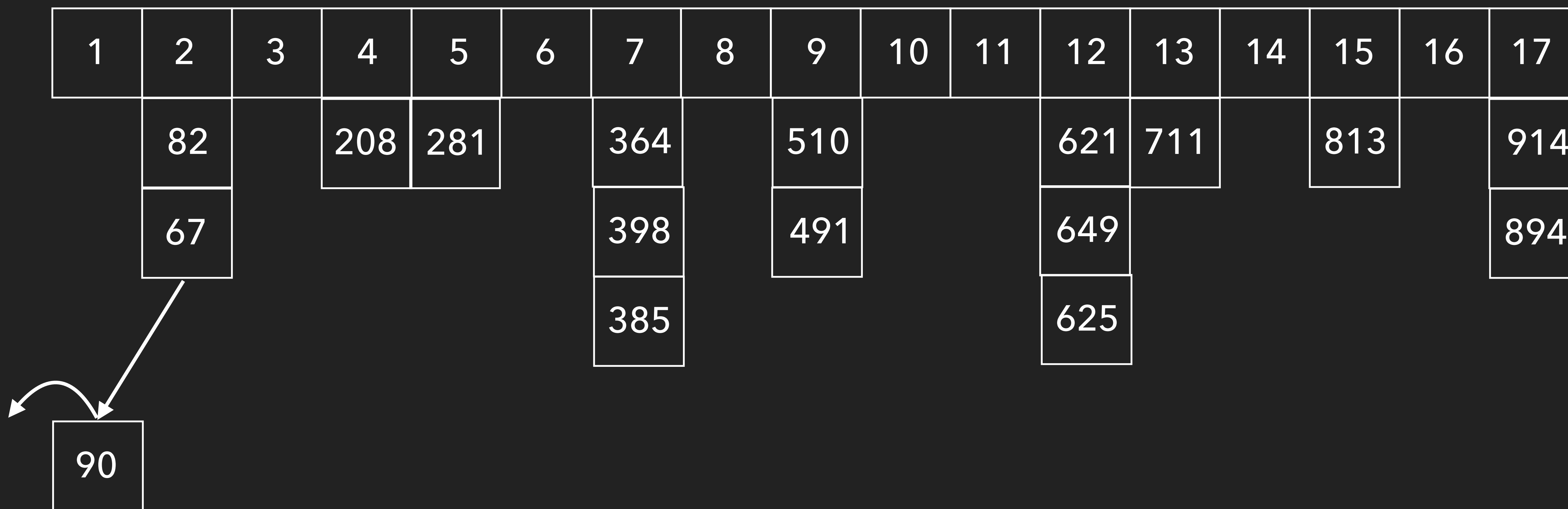
First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

    Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	82		208	281		364		510			621	711		813		914
	67					398		491			649					894
						385					625					
	90															

First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	82		208	281		364		510			621	711		813		914
						398		491			649					894
						385					625					

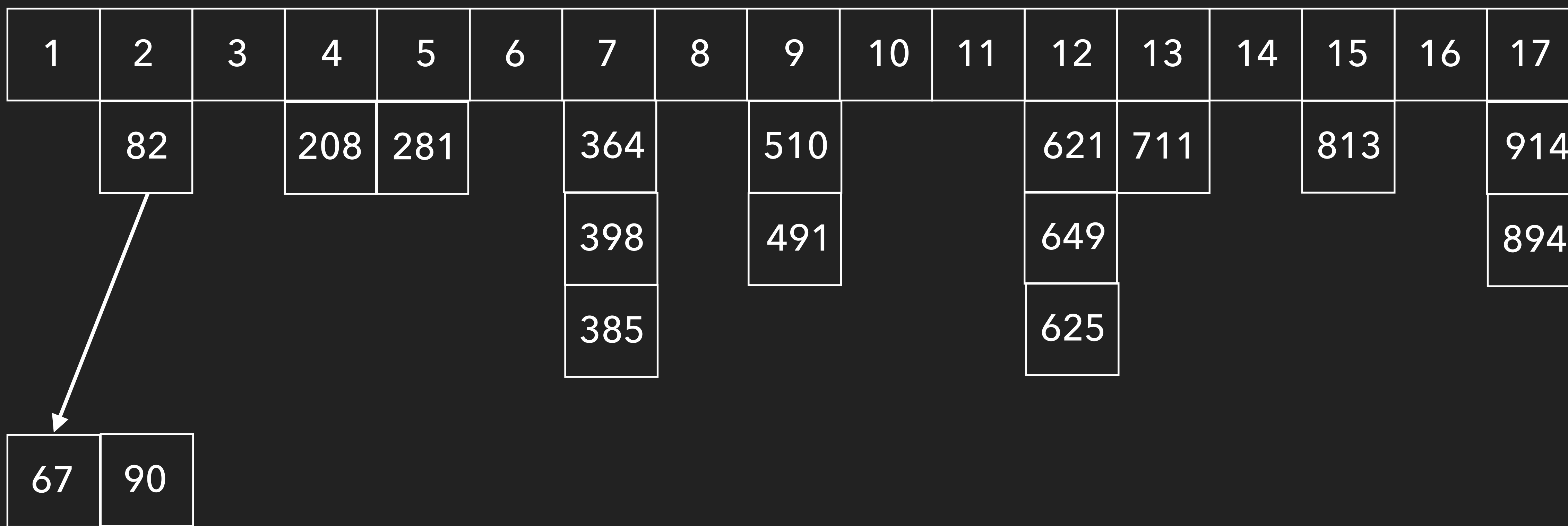
67	90

First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

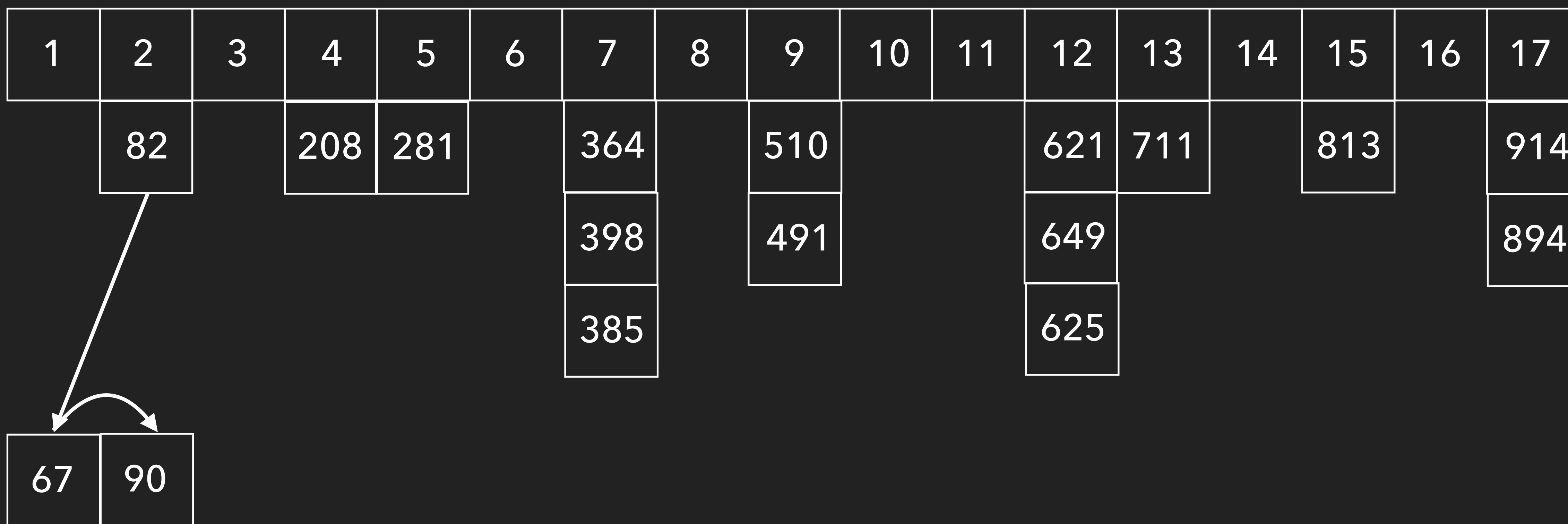


First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	82		208	281		364		510			621	711		813		914
						398		491			649					894
						385					625					

67		90

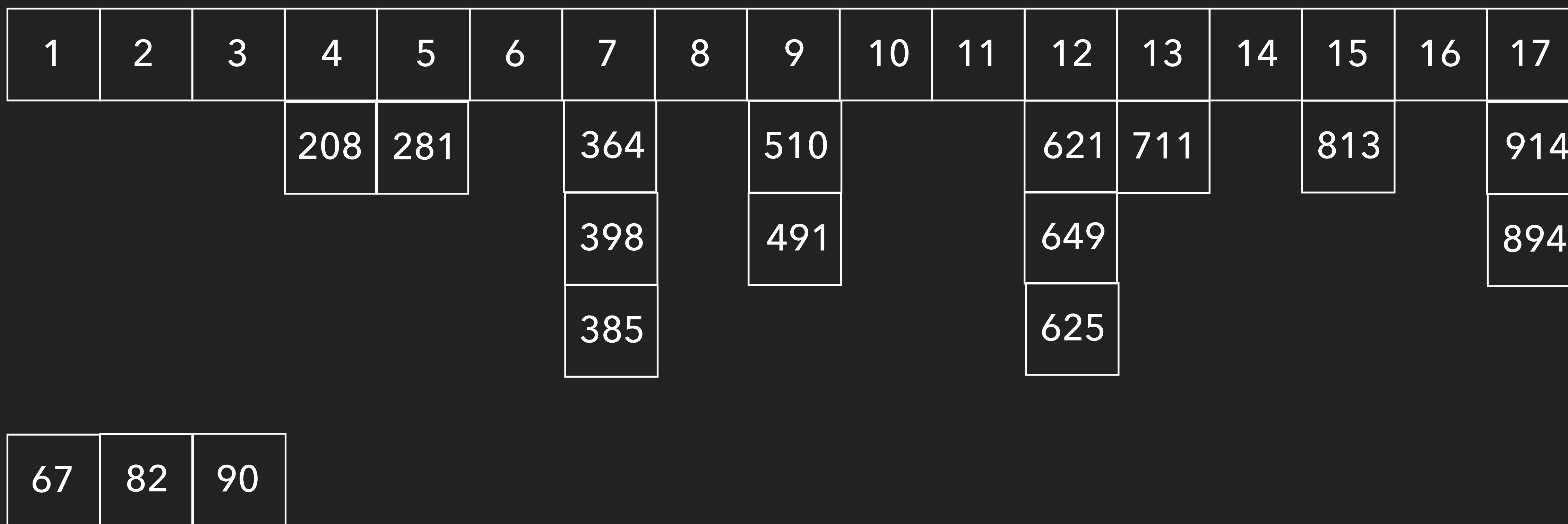
First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

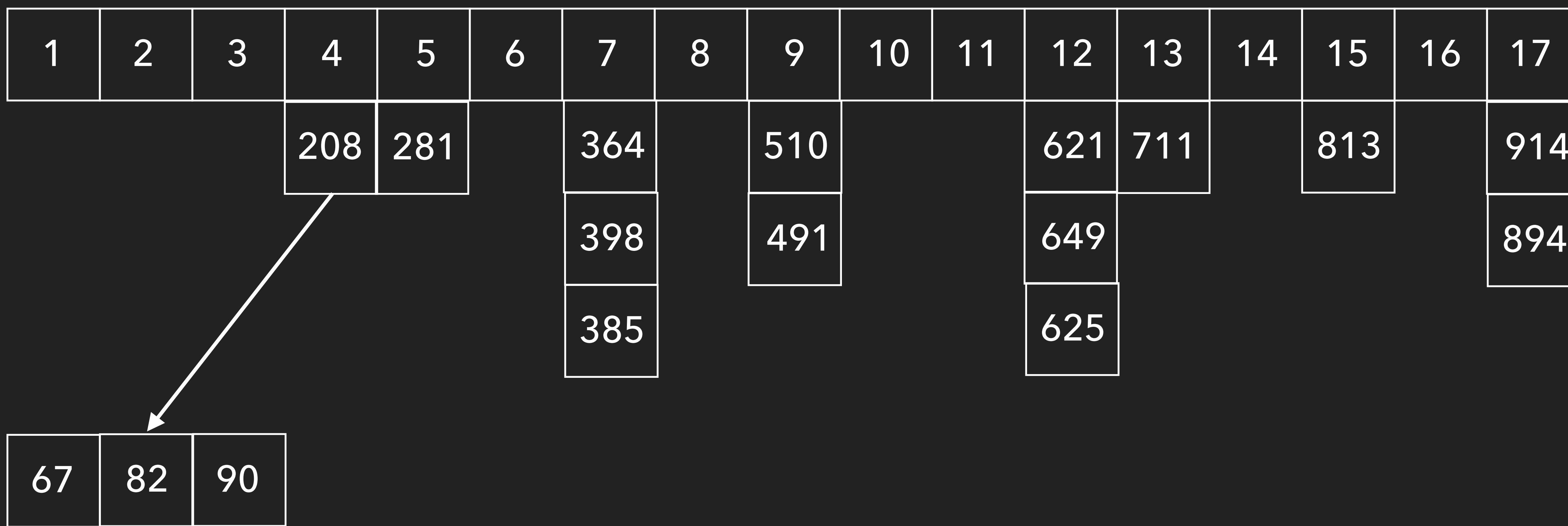


First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

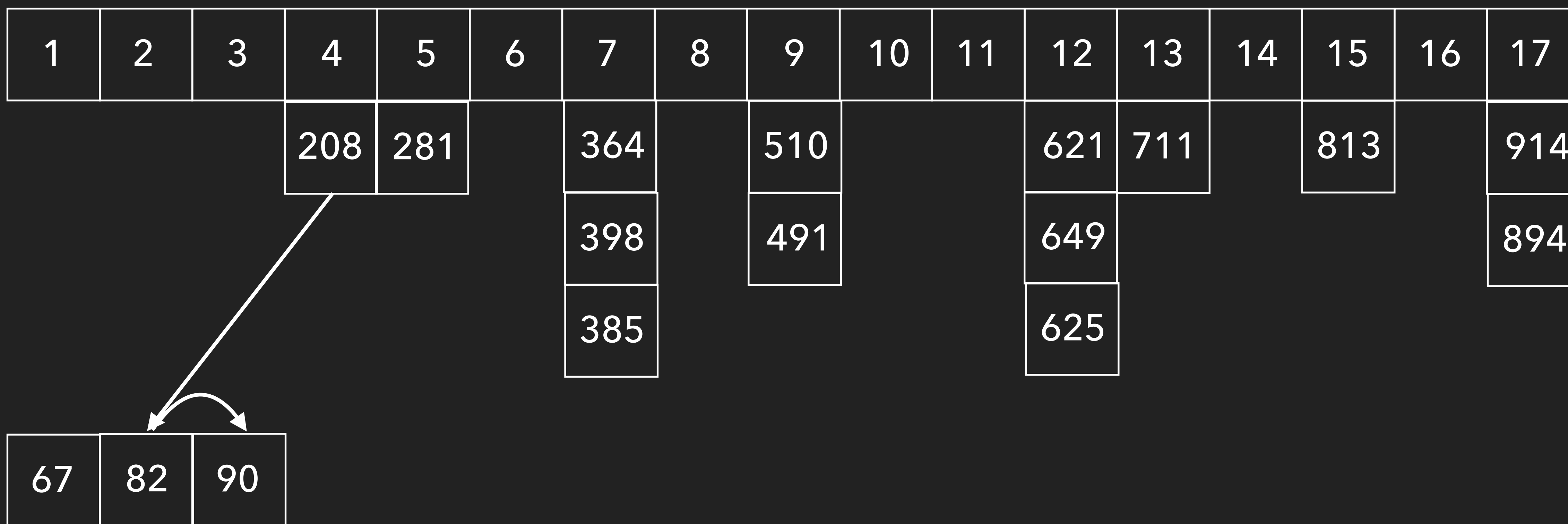


First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

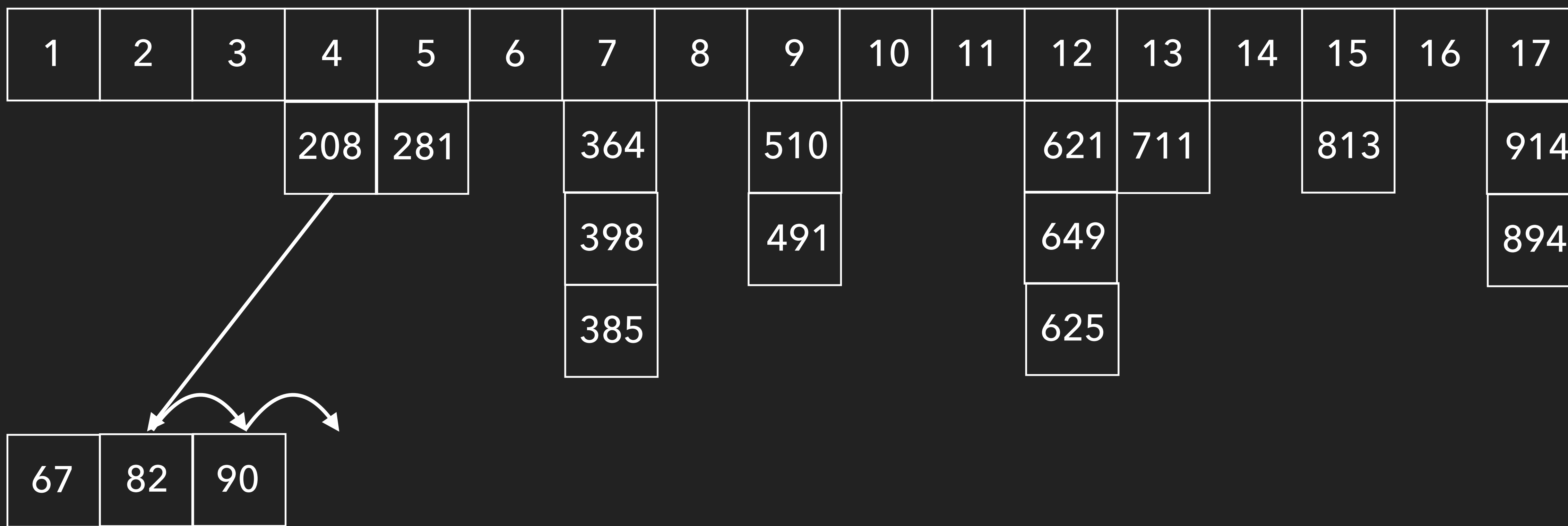


First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
				281		364		510			621	711		813		914
						398		491			649					894
						385					625					

67	82	90	208

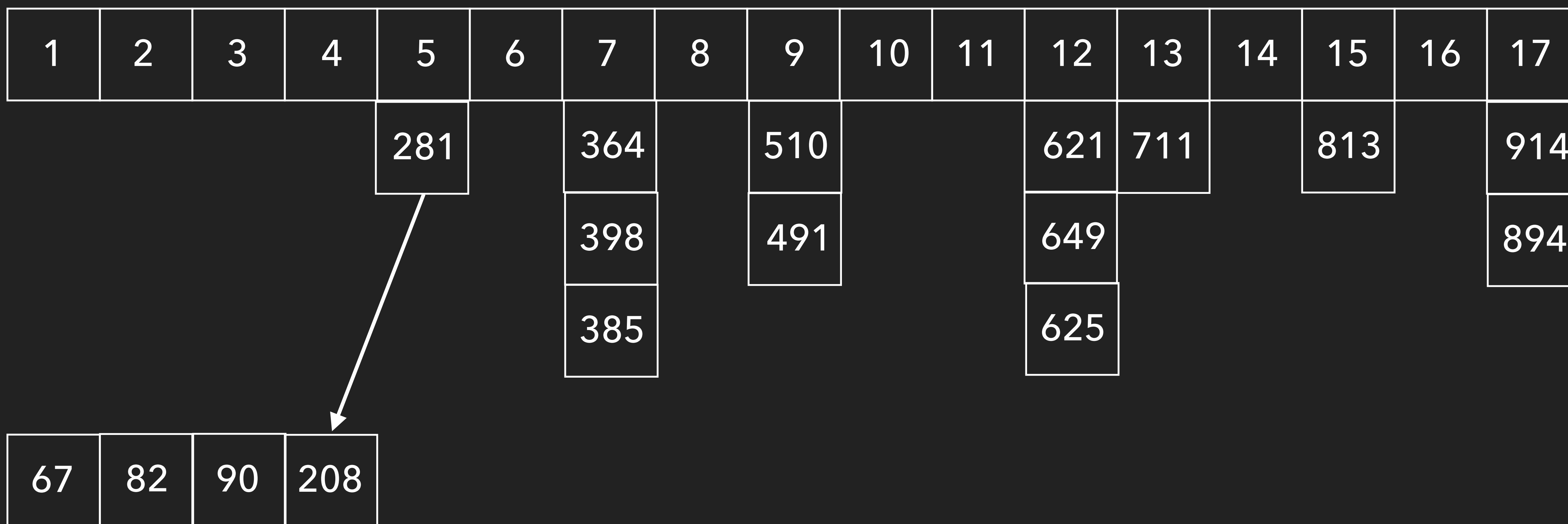
First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

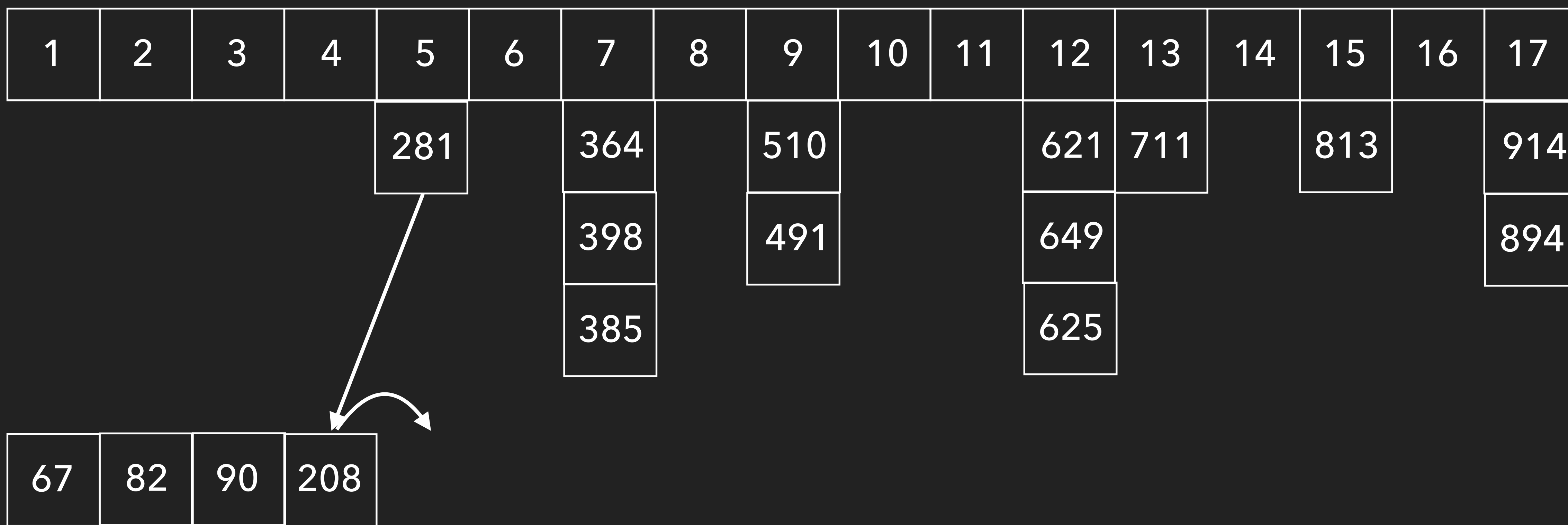


First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						364		510			621	711		813		914
						398		491			649					894
						385					625					

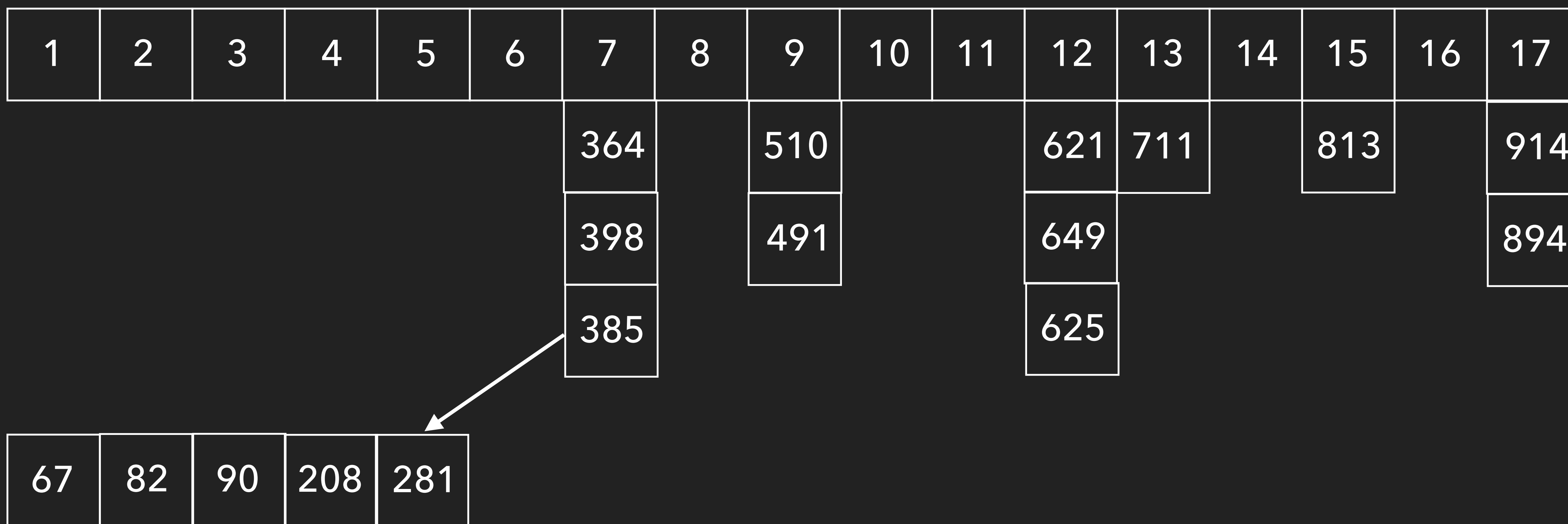
67	82	90	208	281

First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

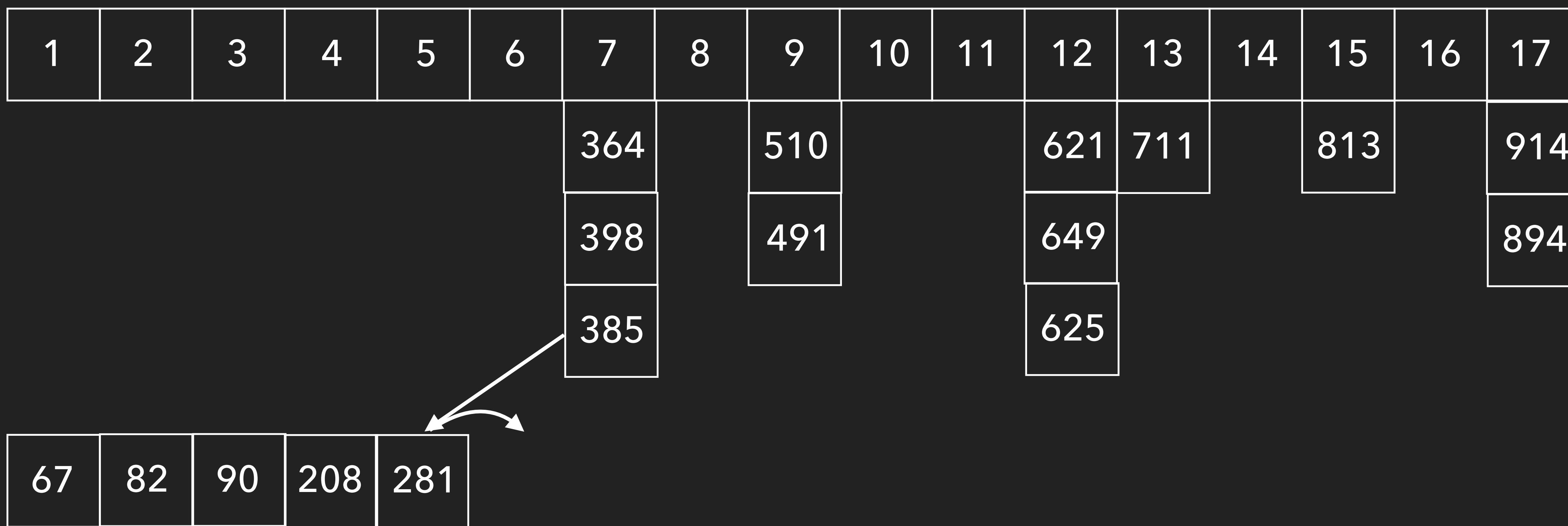


First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						364		510			621	711		813		914
						398		491			649					894
											625					

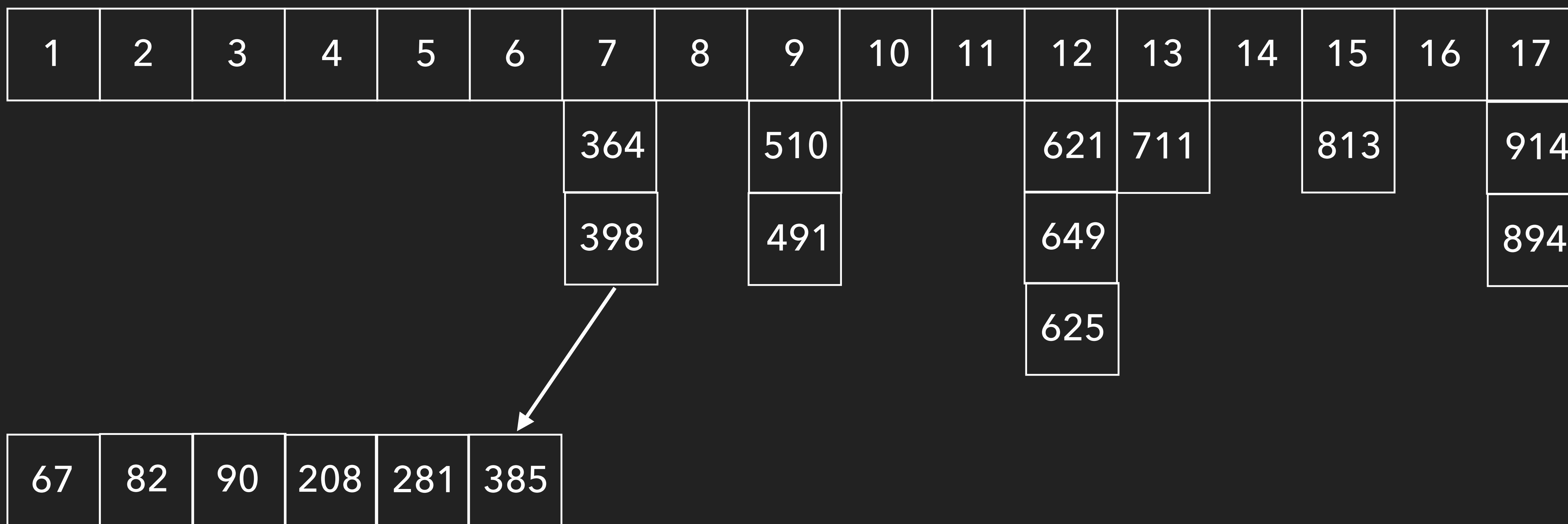
67	82	90	208	281	385

First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



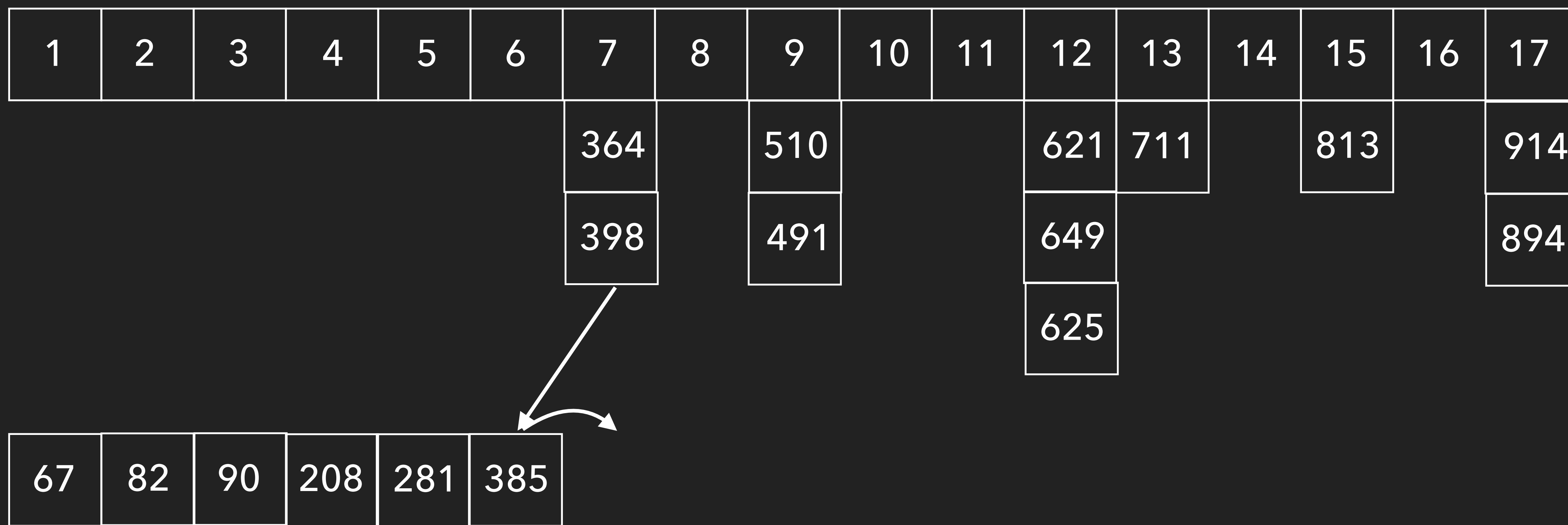
First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

    Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						364		510			621	711		813		914
								491			649					894
											625					

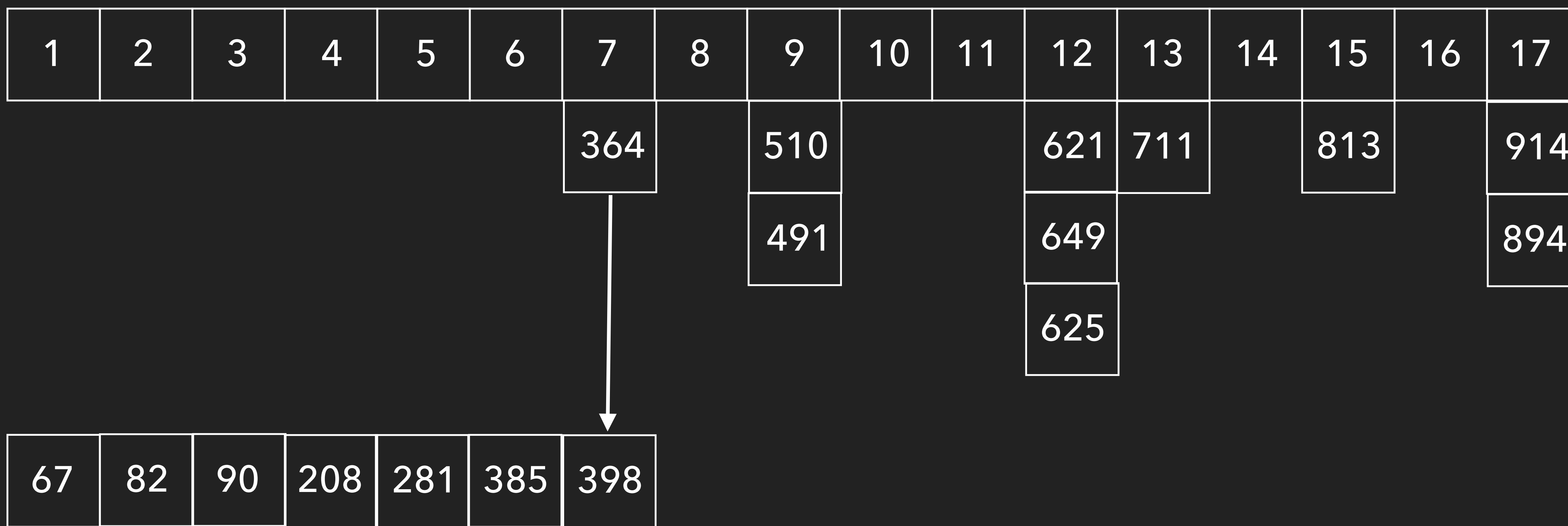
67	82	90	208	281	385	398

First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

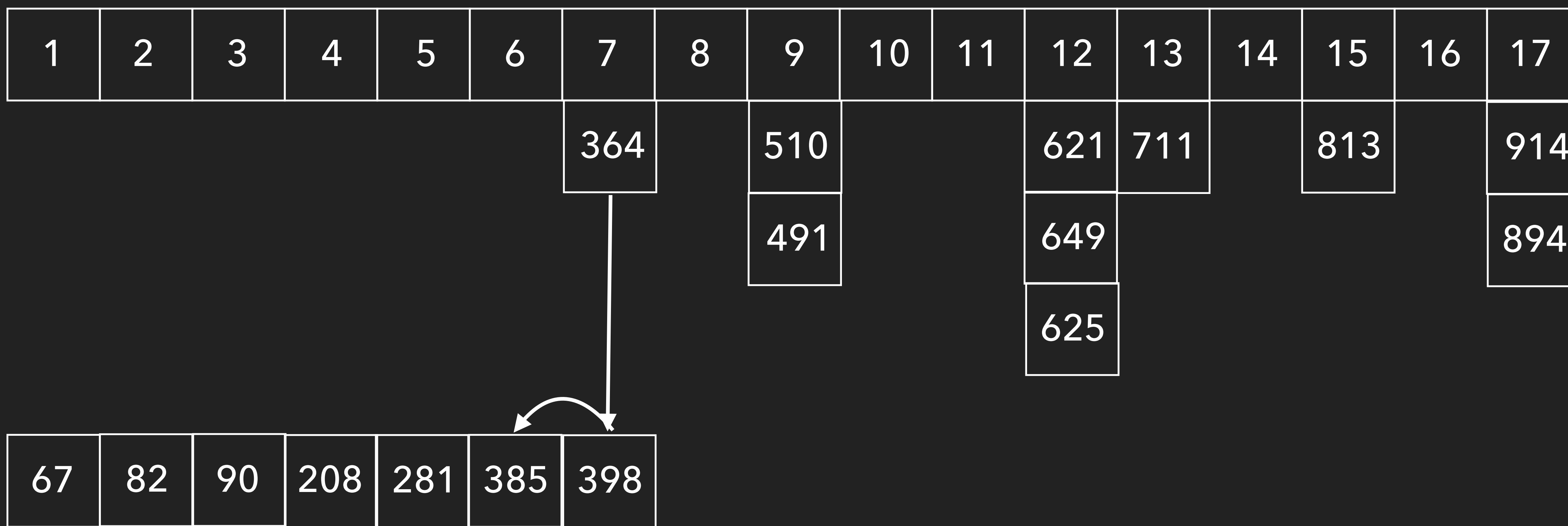


First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

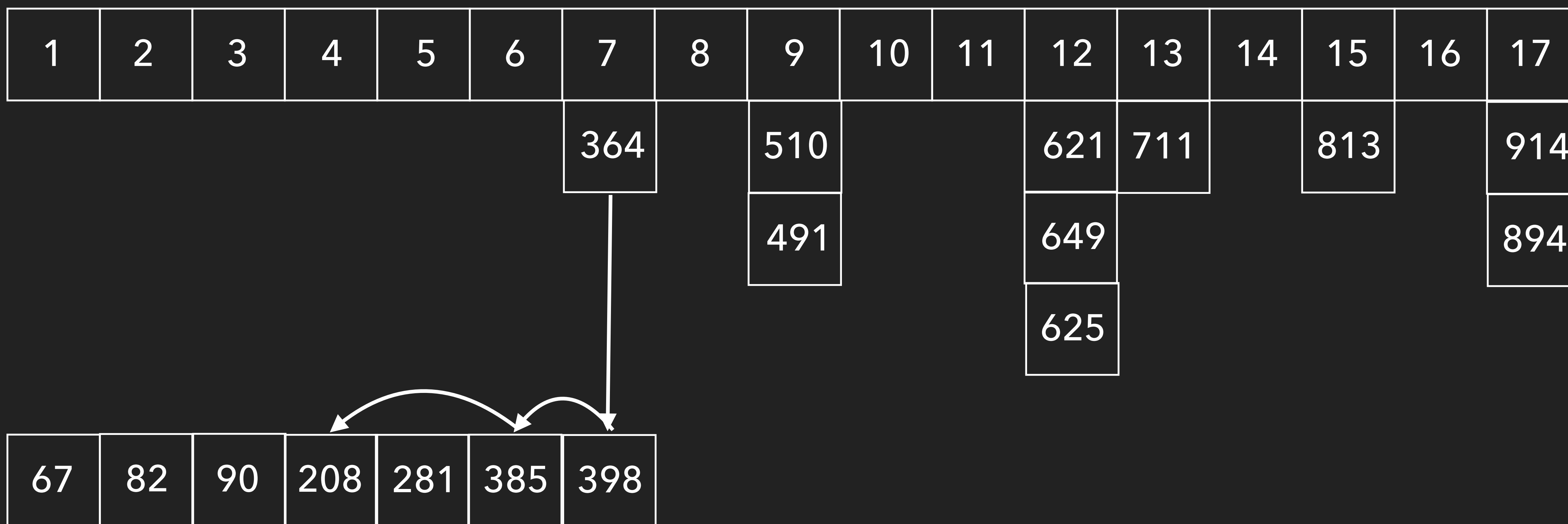


First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



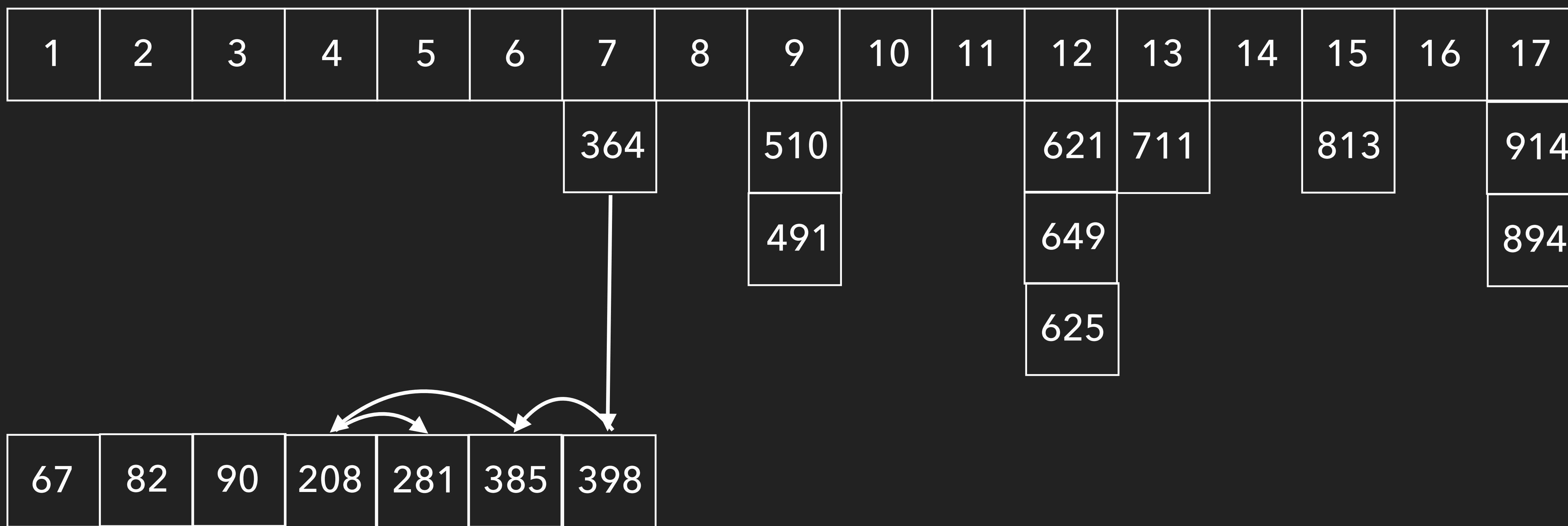
First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12



First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list

   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						364		510			621	711		813		914
								491			649					894
											625					

67	82	90	208	281		385	398

First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

a_i	510	82	208	813	67	491	621	364	914	398	649	281	711	385	90	894	625
$\hat{p}(i)$	9	2	4	15	2	9	12	7	17	7	12	5	13	7	2	17	12

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
								510			621	711		813		914
								491			649					894
											625					

67	82	90	208	281	364	385	398

First algorithm:

1. Bucket sort according to $\hat{p}$

2. From left to right: Insert into sorted list
   Use binary search with predictions to find insert position

WLOG: $\hat{p}(1) \leq \hat{p}(2) \leq \ldots \leq \hat{p}(n)$

#comparisons $= \sum_i \log\big(\,|p(i) - p(i-1)| + 1\big)$

WLOG: $\hat{p}(1) \leq \hat{p}(2) \leq \ldots \leq \hat{p}(n)$

$$\#\text{comparisons} = \sum_i \log\left(\underbrace{|p(i) - p(i-1)|}_{\leq |p(i) - \hat{p}(i)| + \hat{p}(i) - \hat{p}(i-1) + |\hat{p}(i-1) - p(i-1)|} + 1 \right)$$

WLOG: $\hat{p}(1) \leq \hat{p}(2) \leq \ldots \leq \hat{p}(n)$

#comparisons $= \sum_i \log\left(\underbrace{|p(i) - p(i-1)|} + 1\right)$

$$\leq \underbrace{|p(i) - \hat{p}(i)|}_{\eta_i} + \hat{p}(i) - \hat{p}(i-1) + \underbrace{|\hat{p}(i-1) - p(i-1)|}_{\eta_{i-1}}$$

WLOG: $\hat{p}(1) \le \hat{p}(2) \le \ldots \le \hat{p}(n)$

$$\#\text{comparisons} = \sum_i \log\left(\underbrace{|p(i) - p(i-1)|} + 1 \right)$$

$$\le \underbrace{|p(i) - \hat{p}(i)|}_{\eta_i} + \hat{p}(i) - \hat{p}(i-1) + \underbrace{|\hat{p}(i-1) - p(i-1)|}_{\eta_{i-1}}$$

$$\le \sum_i O(\log \eta_i) + \sum_i \log\left(\hat{p}(i) - \hat{p}(i-1) + 1\right)$$

WLOG: $\hat{p}(1) \le \hat{p}(2) \le \ldots \le \hat{p}(n)$

$$\#\text{comparisons} = \sum_i \log\left(\underbrace{|p(i) - p(i-1)|}_{\le \underbrace{|p(i)-\hat{p}(i)|}_{\eta_i} + \hat{p}(i)-\hat{p}(i-1) + \underbrace{|\hat{p}(i-1)-p(i-1)|}_{\eta_{i-1}}} + 1 \right)$$

$$\le \sum_i O(\log \eta_i) + \sum_i \underbrace{\log\left(\hat{p}(i) - \hat{p}(i-1) + 1\right)}_{\le \hat{p}(i)-\hat{p}(i-1)}$$

WLOG: $\hat{p}(1) \leq \hat{p}(2) \leq \ldots \leq \hat{p}(n)$

$$\#\text{comparisons} = \sum_i \log\left(\underbrace{|p(i) - p(i-1)|}_{\leq \underbrace{|p(i) - \hat{p}(i)|}_{\eta_i} + \hat{p}(i) - \hat{p}(i-1) + \underbrace{|\hat{p}(i-1) - p(i-1)|}_{\eta_{i-1}}} + 1 \right)$$

$$\leq \sum_i O(\log \eta_i) + \underbrace{\sum_i \underbrace{\log\left(\hat{p}(i) - \hat{p}(i-1) + 1\right)}_{\leq \hat{p}(i) - \hat{p}(i-1)}}_{\leq \hat{p}(n) - \hat{p}(1) \leq n}$$

WLOG: $\hat{p}(1) \leq \hat{p}(2) \leq \ldots \leq \hat{p}(n)$

$$\#\text{comparisons} = \sum_i \log\left(\underbrace{|p(i) - p(i-1)|}_{\leq \underbrace{|p(i)-\hat{p}(i)|}_{\eta_i} + \hat{p}(i)-\hat{p}(i-1) + \underbrace{|\hat{p}(i-1)-p(i-1)|}_{\eta_{i-1}}} + 1\right)$$

$$\leq \sum_i O(\log \eta_i) + \underbrace{\sum_i \underbrace{\log\left(\hat{p}(i) - \hat{p}(i-1) + 1\right)}_{\leq \hat{p}(i)-\hat{p}(i-1)}}_{\leq \hat{p}(n)-\hat{p}(1) \leq n}$$

$$\leq O\left(\sum_{i=1}^{n} \log(\eta_i + 2)\right)$$

But shifting subarrays slow

But shifting subarrays slow

Better: Replace array by BBST to get time $O\left(\sum_i \log(\eta_i + 2)\right)$

More involved algorithm (see our paper [Bai,Coester 23])

$\implies O\left(\sum_i \log(\tilde{\eta}_i + 2)\right)$ comparisons, where

$$\begin{aligned}\tilde{\eta}_i := \min\Big\{&\#\{j\colon a_j < a_i, \hat{p}(j) \geq \hat{p}(i)\},\\ &\#\{j\colon a_j > a_i, \hat{p}(j) \leq \hat{p}(i)\}\Big\}\end{aligned}$$

Sorting with Dirty and Clean Comparisons

Input: $a_1, a_2, \ldots, a_n$

slow-and-clean comparator $<$

quick-and-dirty comparator $\hat{<}$

Error: $\eta_i := \#\{j\colon (a_j < a_i) \neq (a_j \mathrel{\hat{<}} a_i)\}$

Sorting with Dirty and Clean Comparisons

Input: $a_1, a_2, \ldots, a_n$

slow-and-clean comparator $<$

quick-and-dirty comparator $\hat{<}$

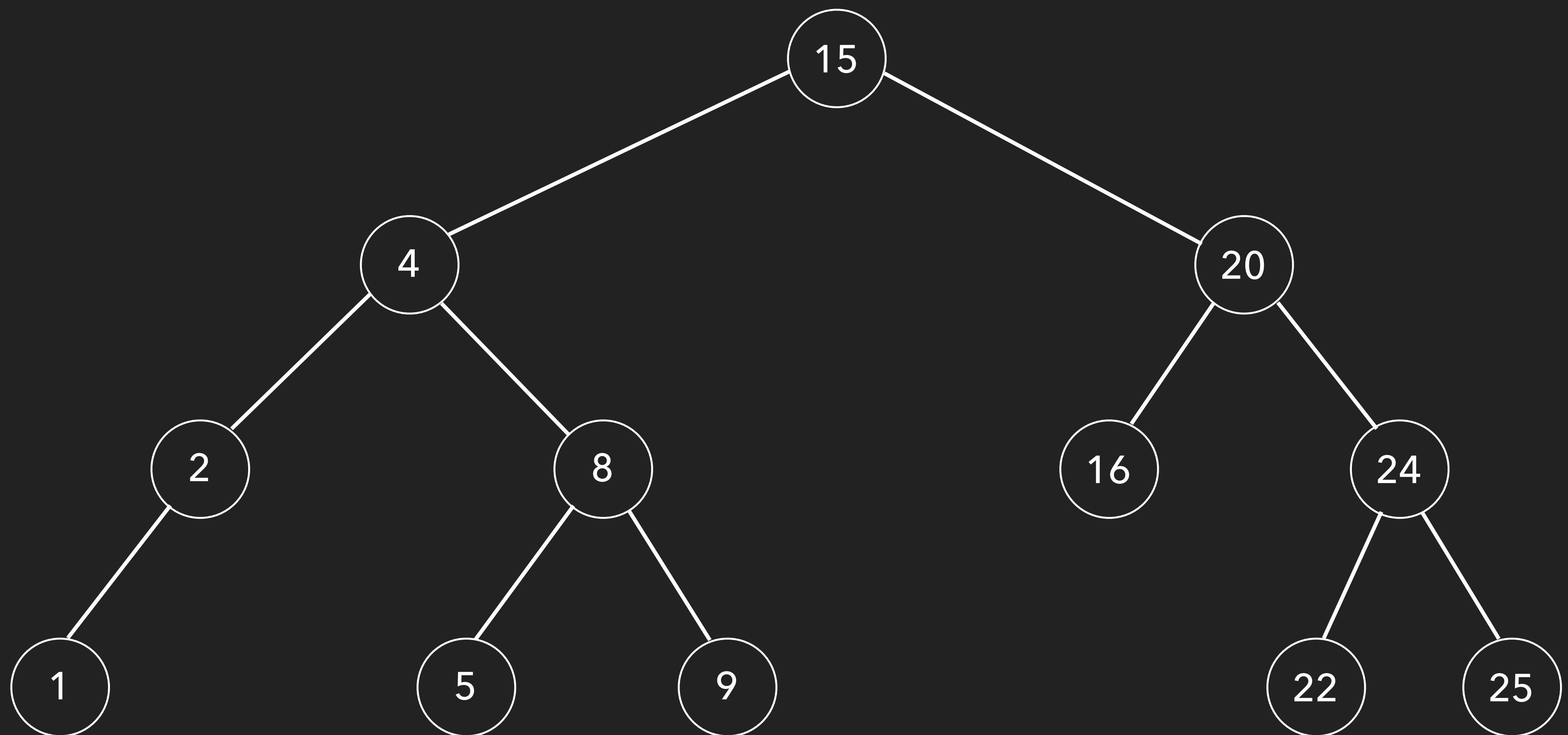
Error: $\eta_i := \#\{j\colon (a_j < a_i) \neq (a_j \mathrel{\hat{<}} a_i)\}$

Theorem: Can sort with $O(n \log n)$ dirty comparisons

and $O\left(\sum_{i=1}^{n} \log(\eta_i + 2)\right)$ clean comparisons

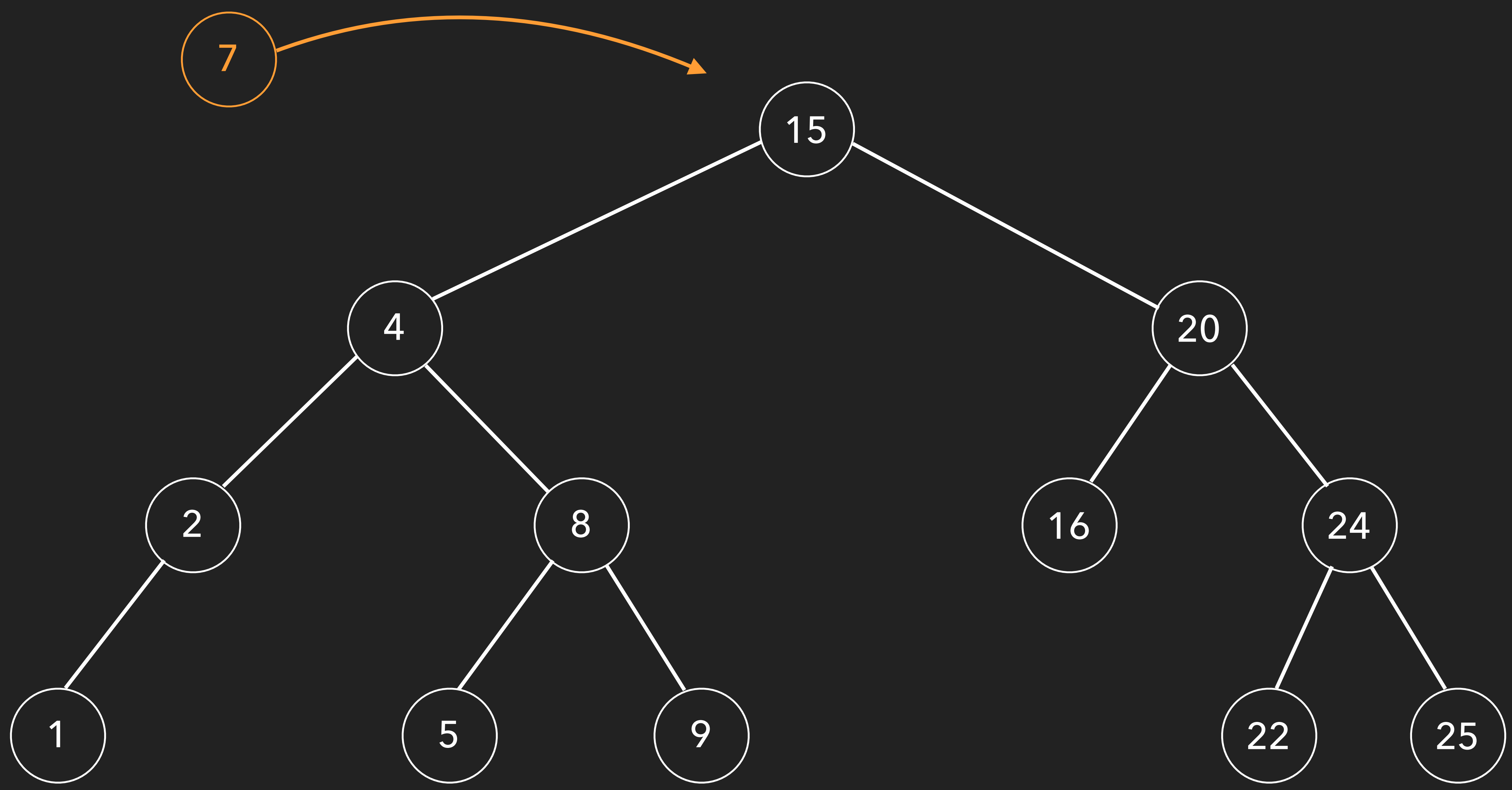
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



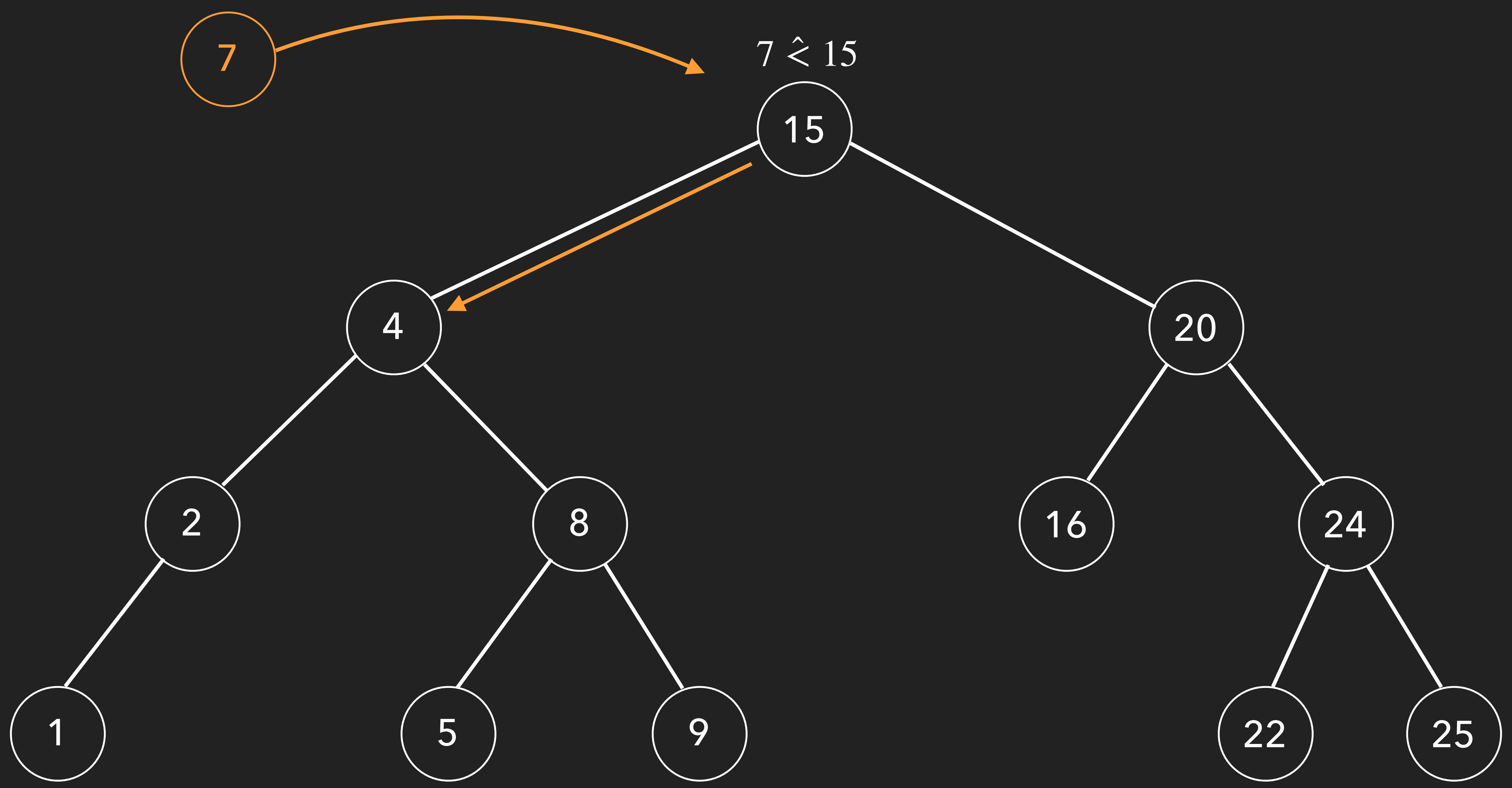
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



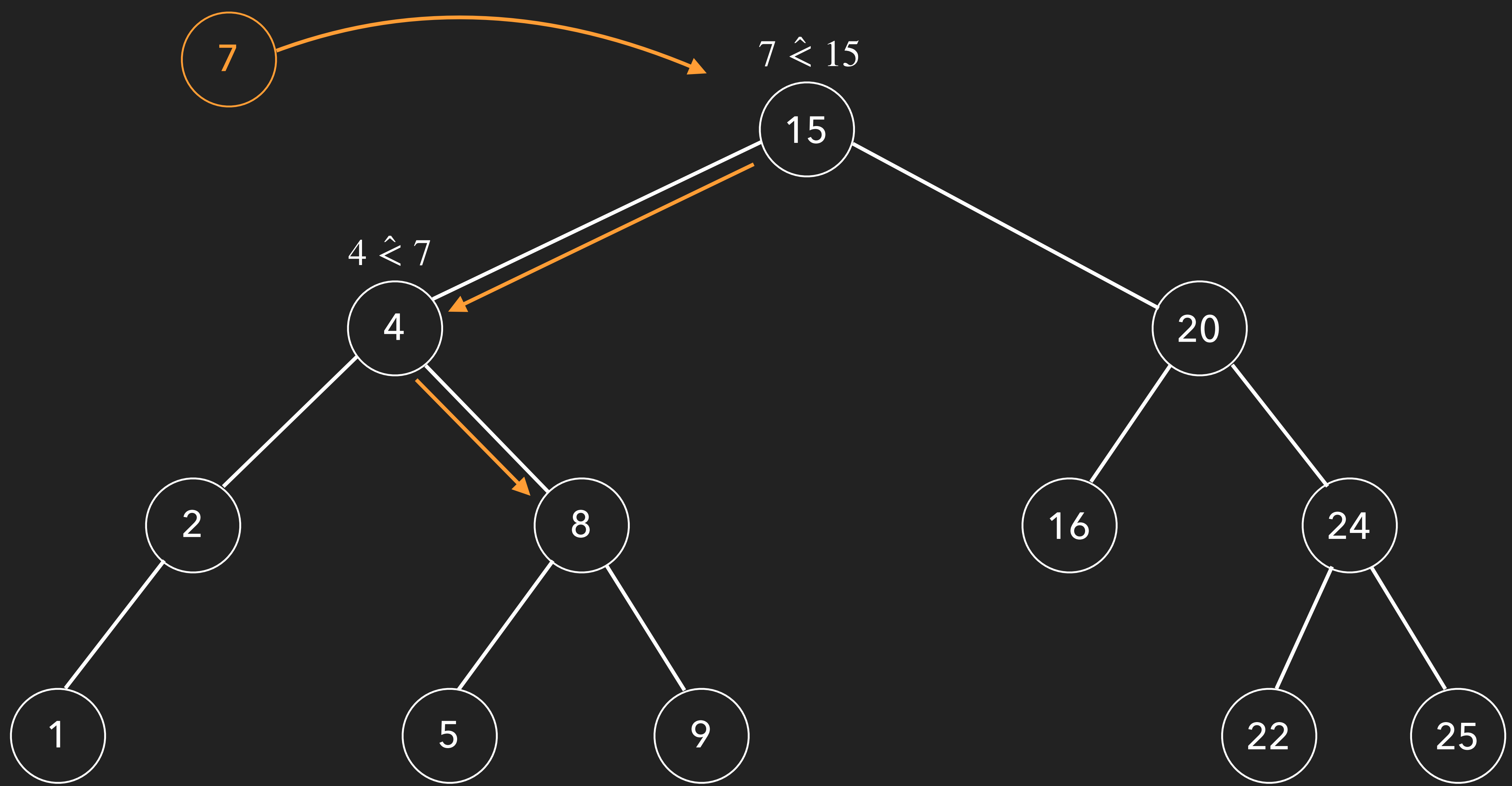
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



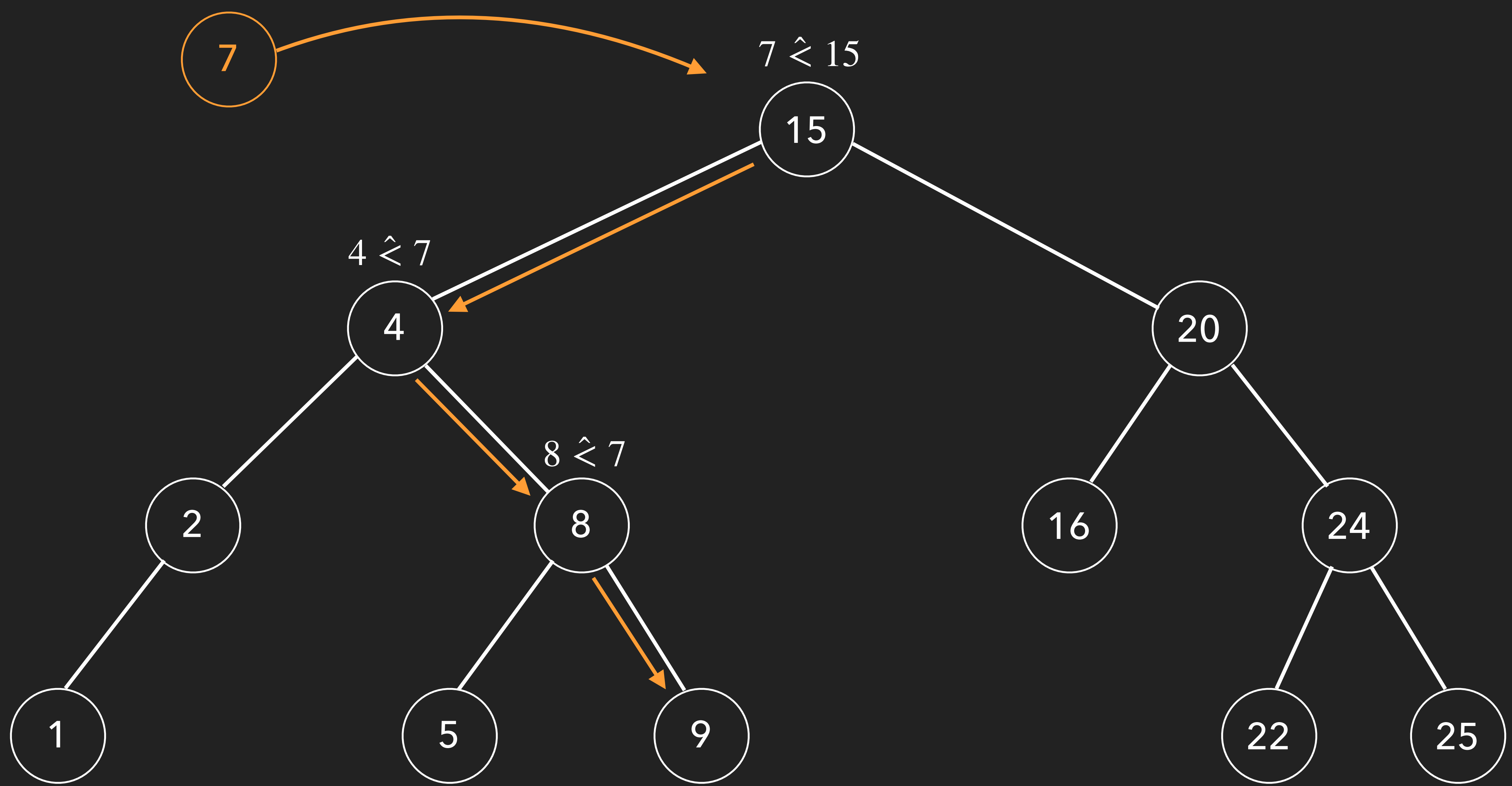
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



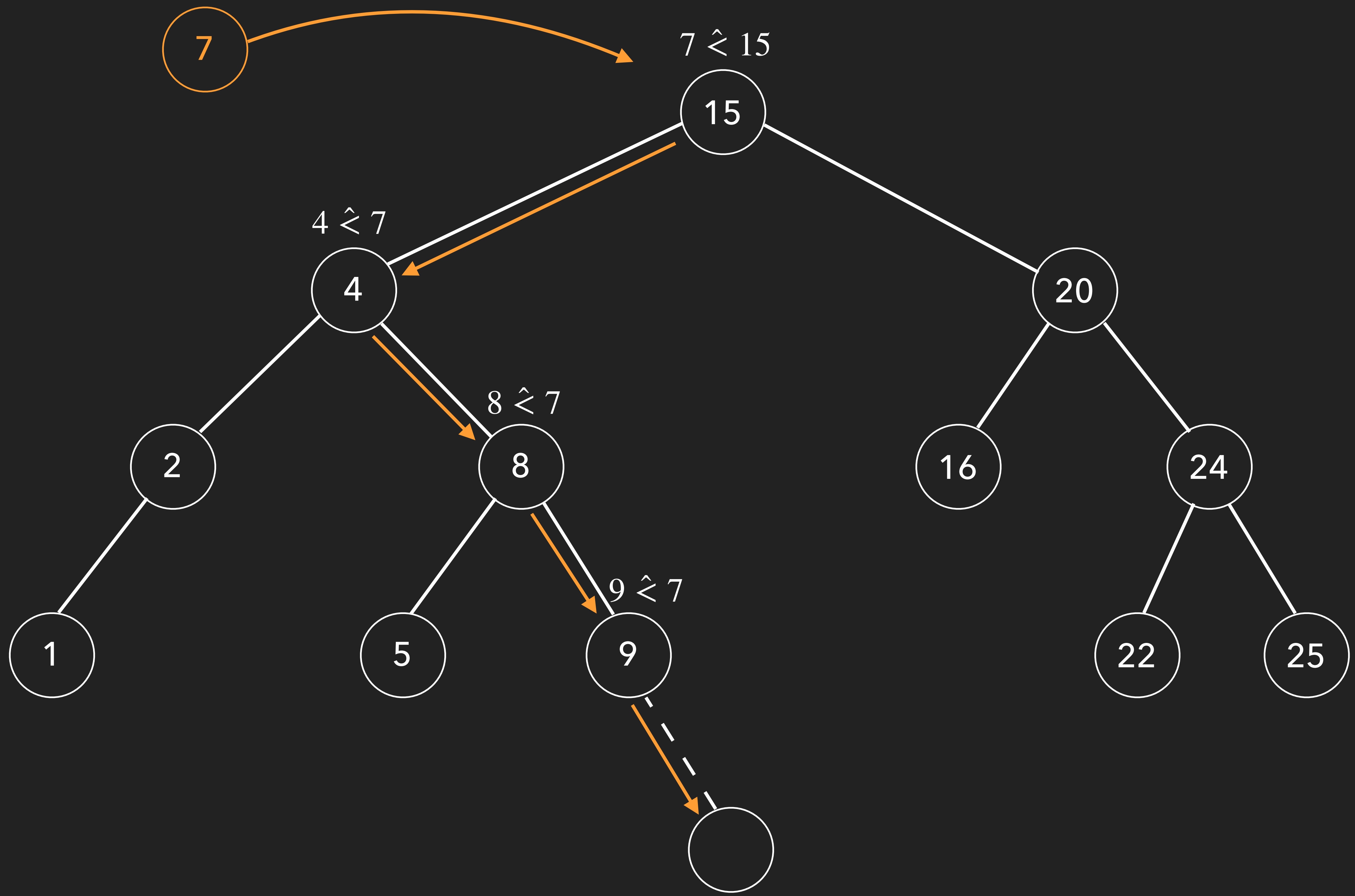
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



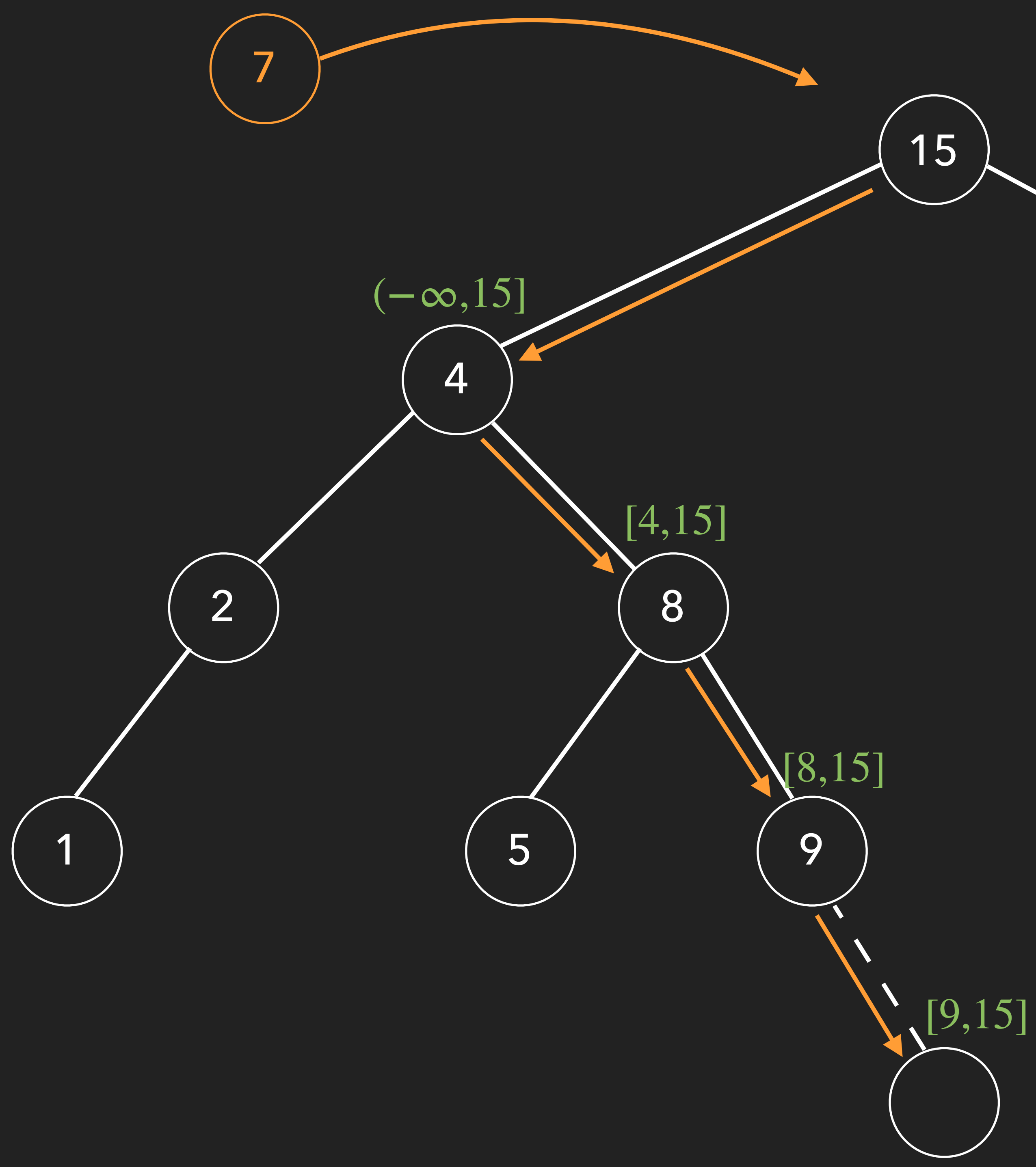
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



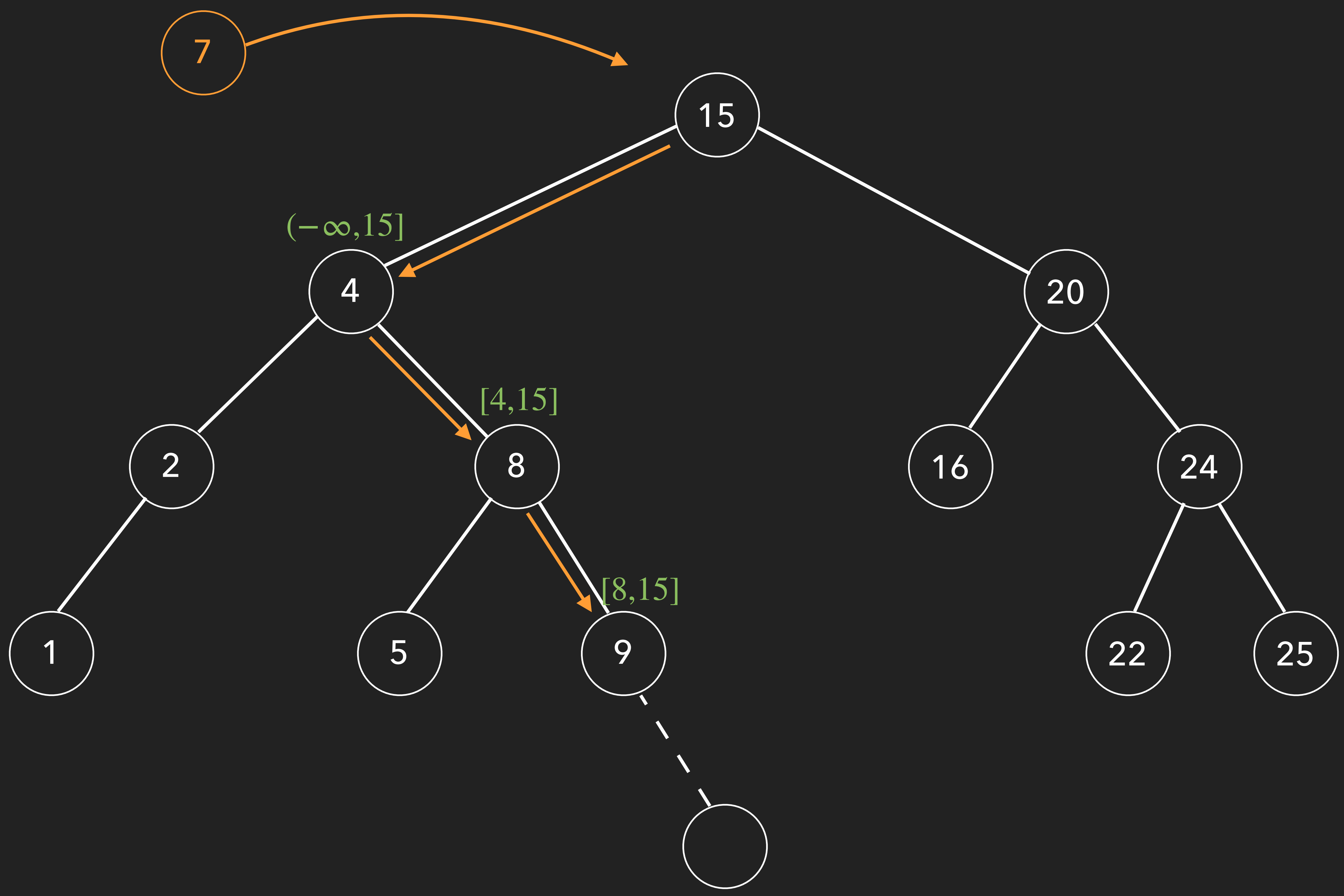
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



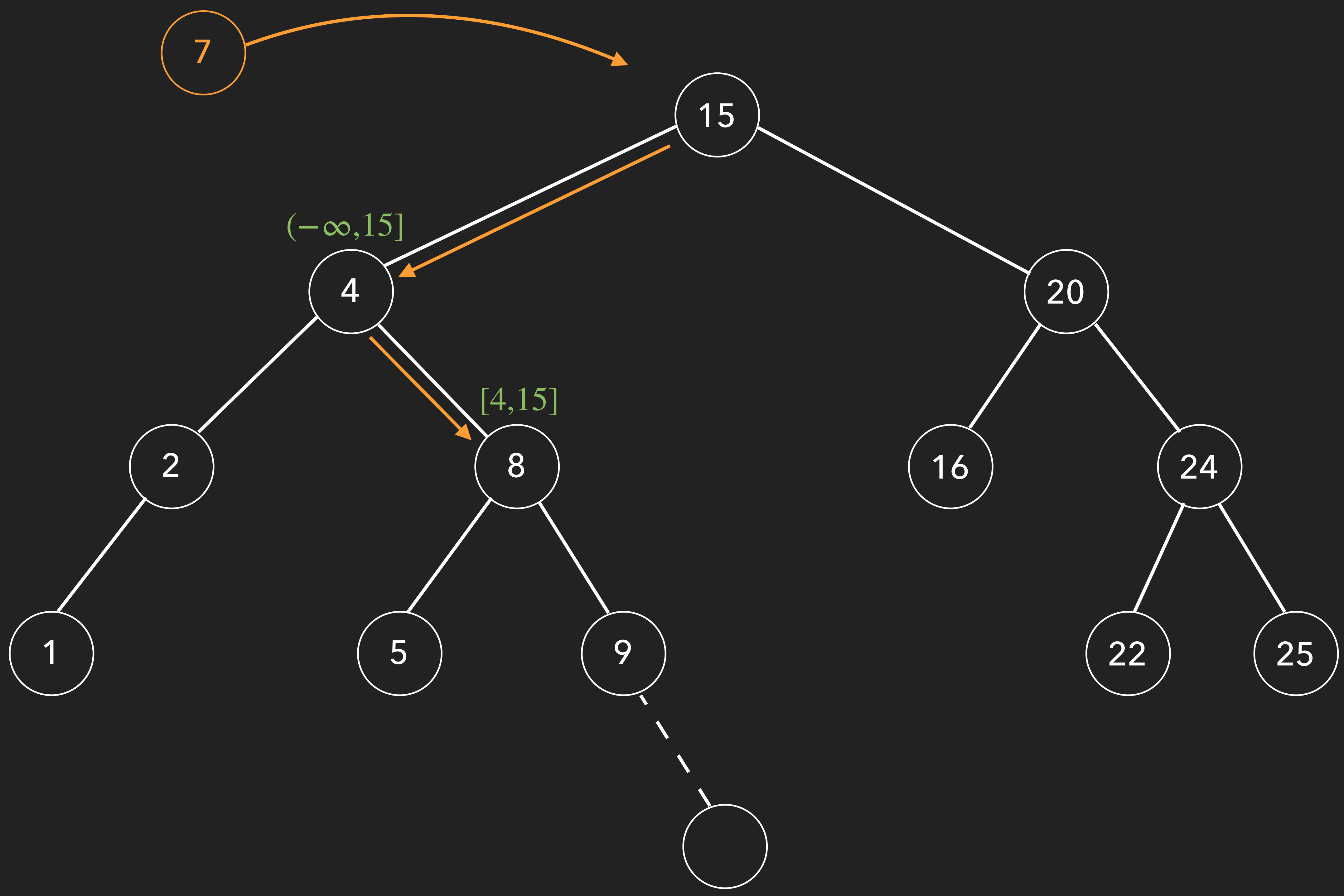
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



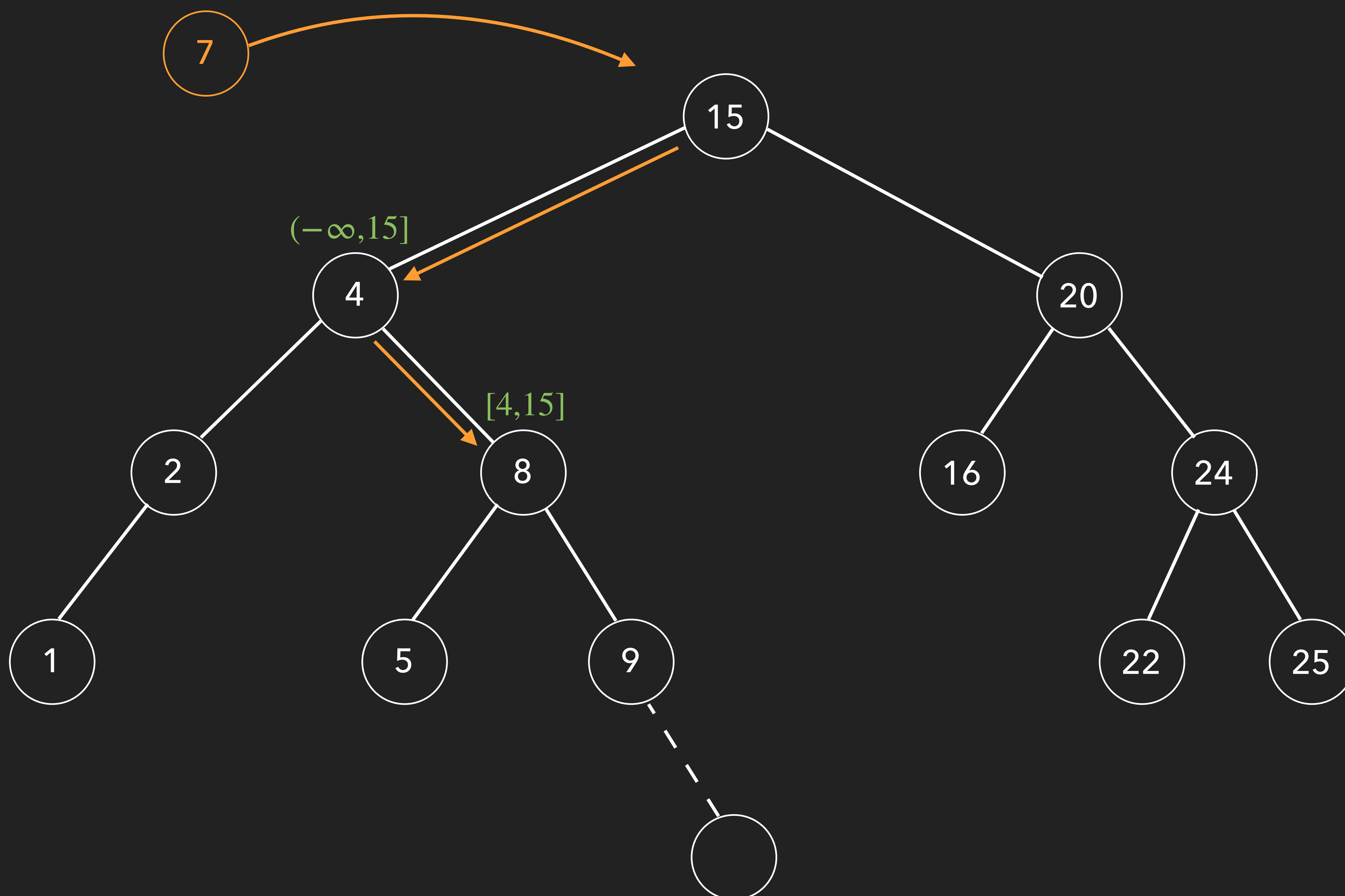
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



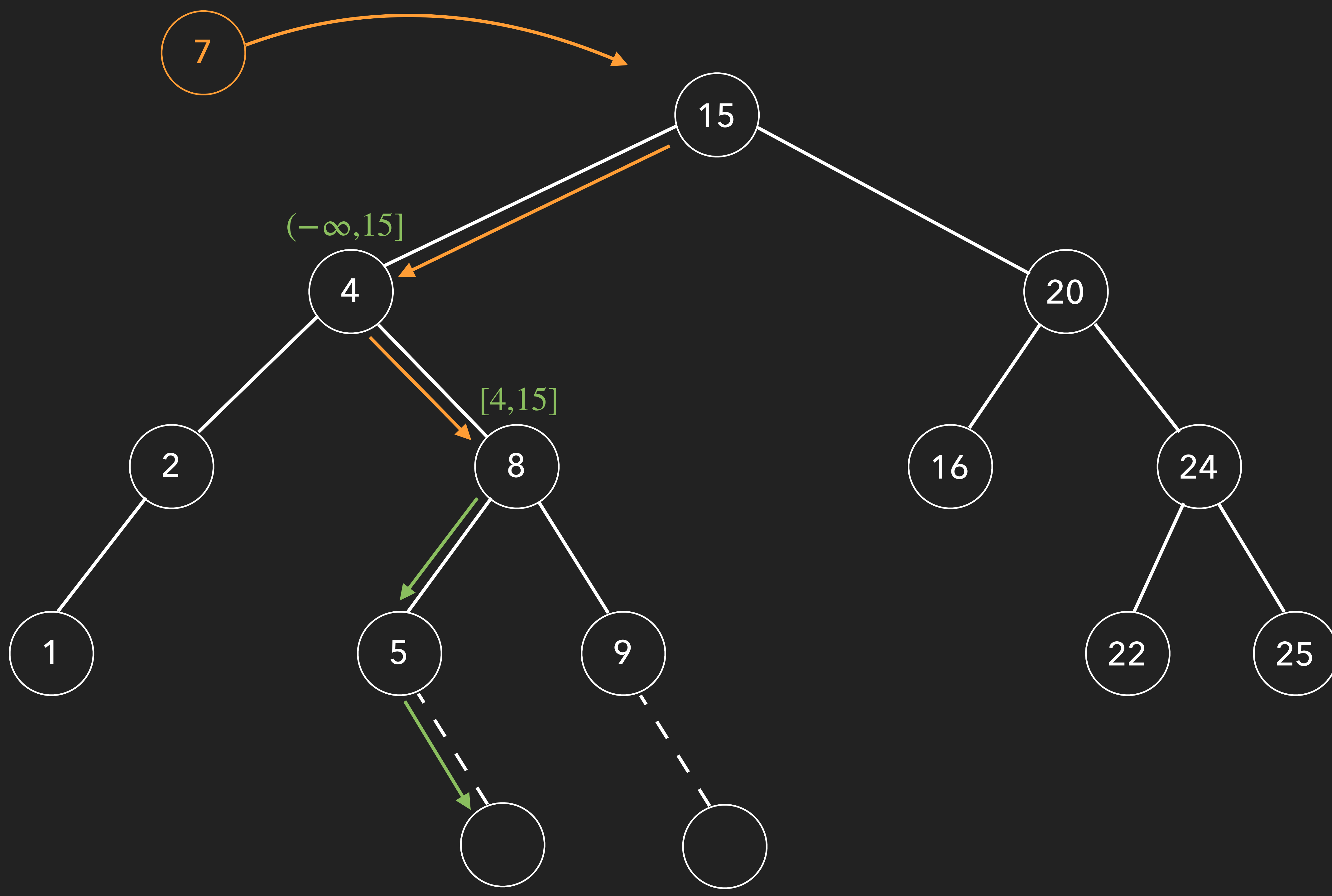
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



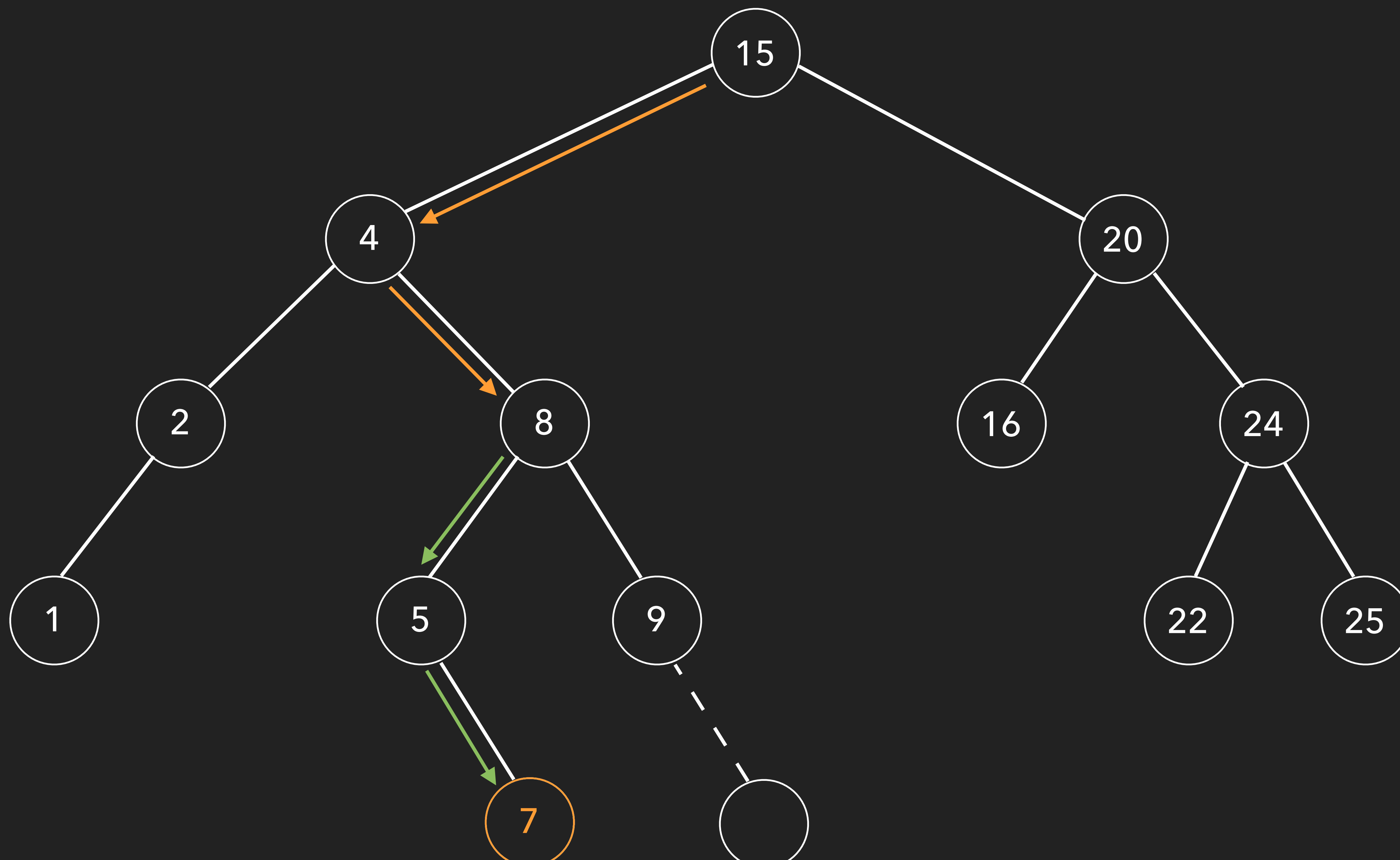
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



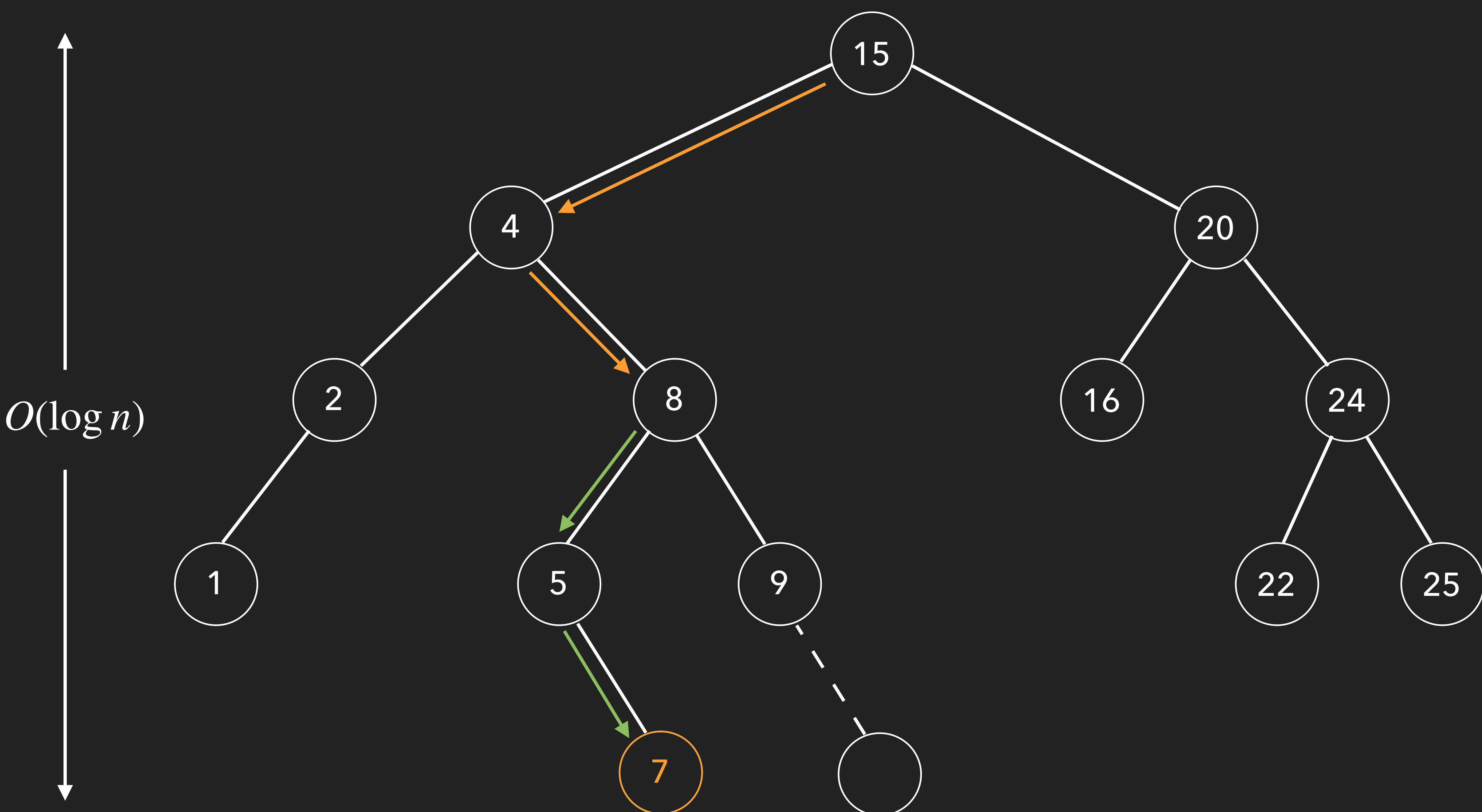
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



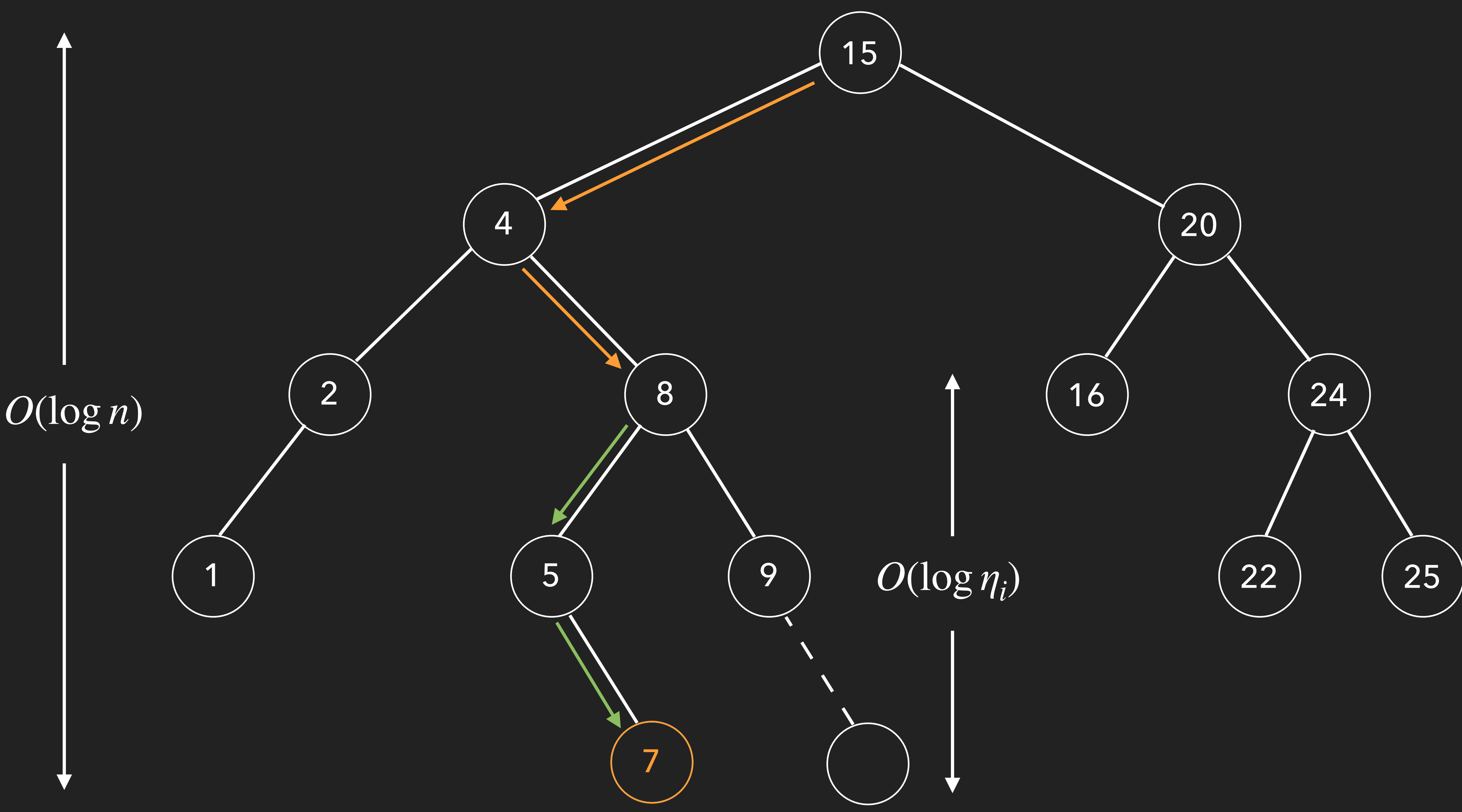
Idea: Build BST wrt. $<$

Guide insertions via $\hat{<}$ and $<$



Idea: Build BST wrt. $<$

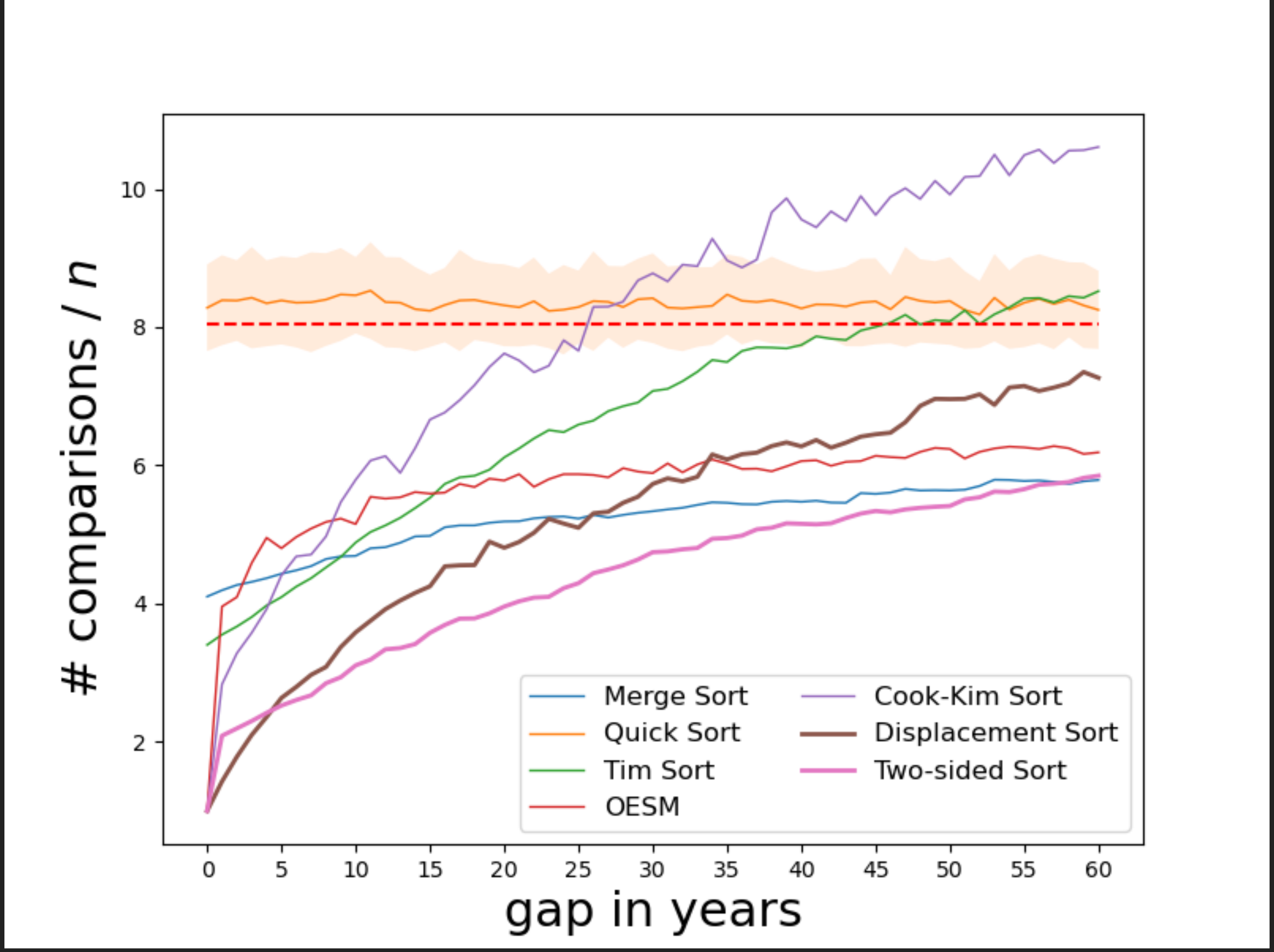
Guide insertions via $\hat{<}$ and $<$



Experiments

Sorting countries by population (n=261)

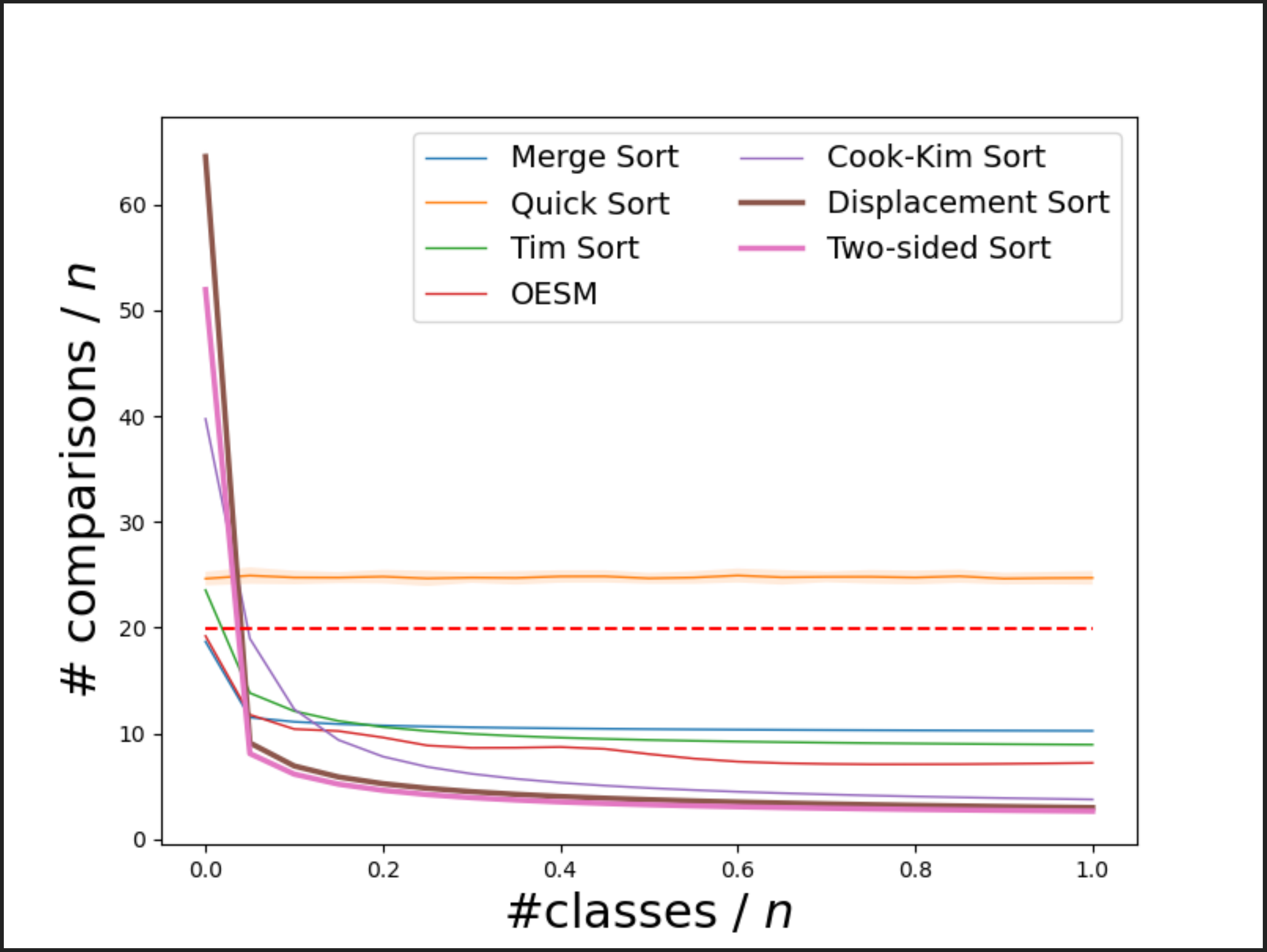
Predictions: ranking x years ago



Experiments

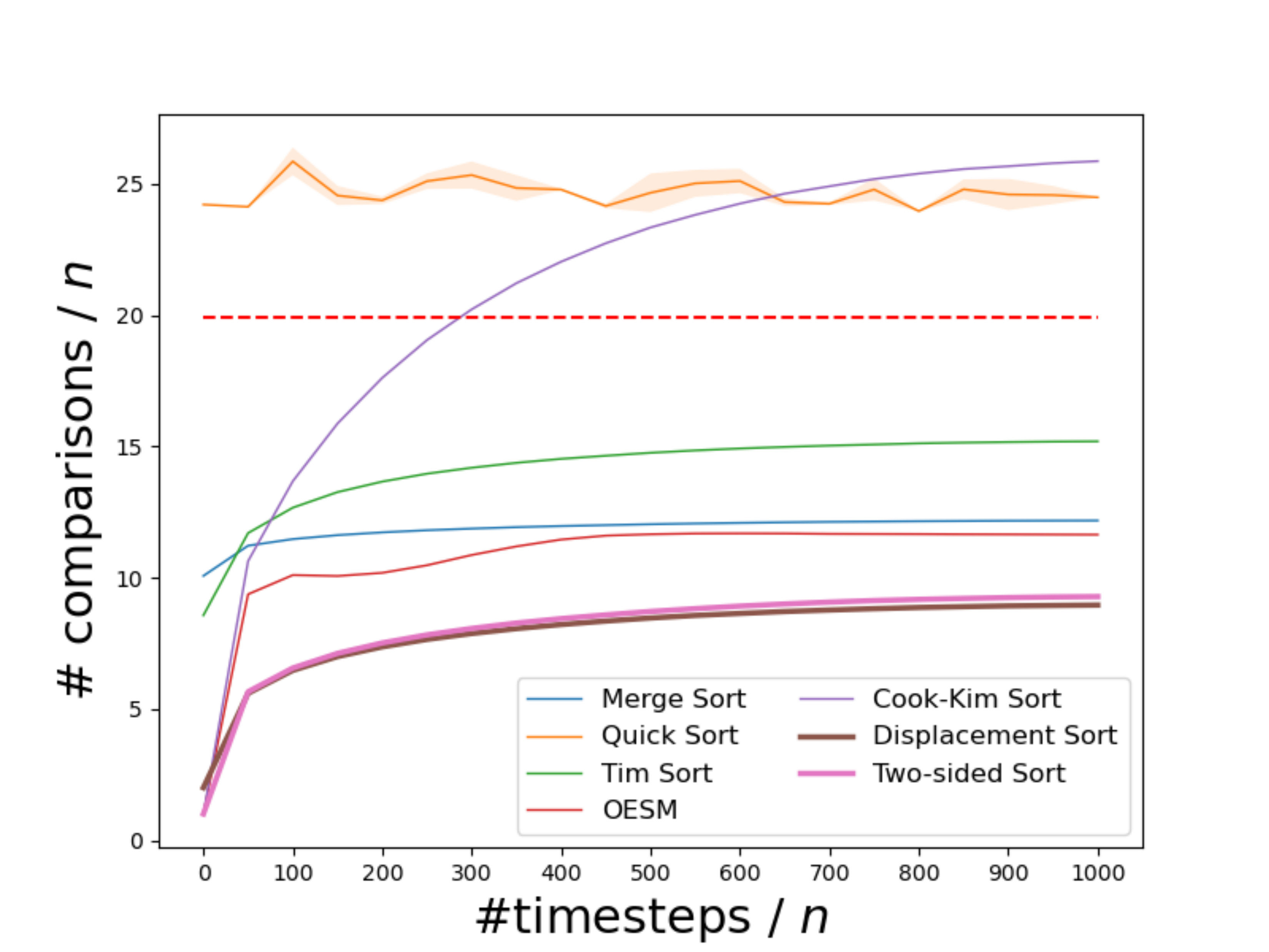
Classes of consecutive items (n=1,000,000)

Predictions: random position within class



Experiments

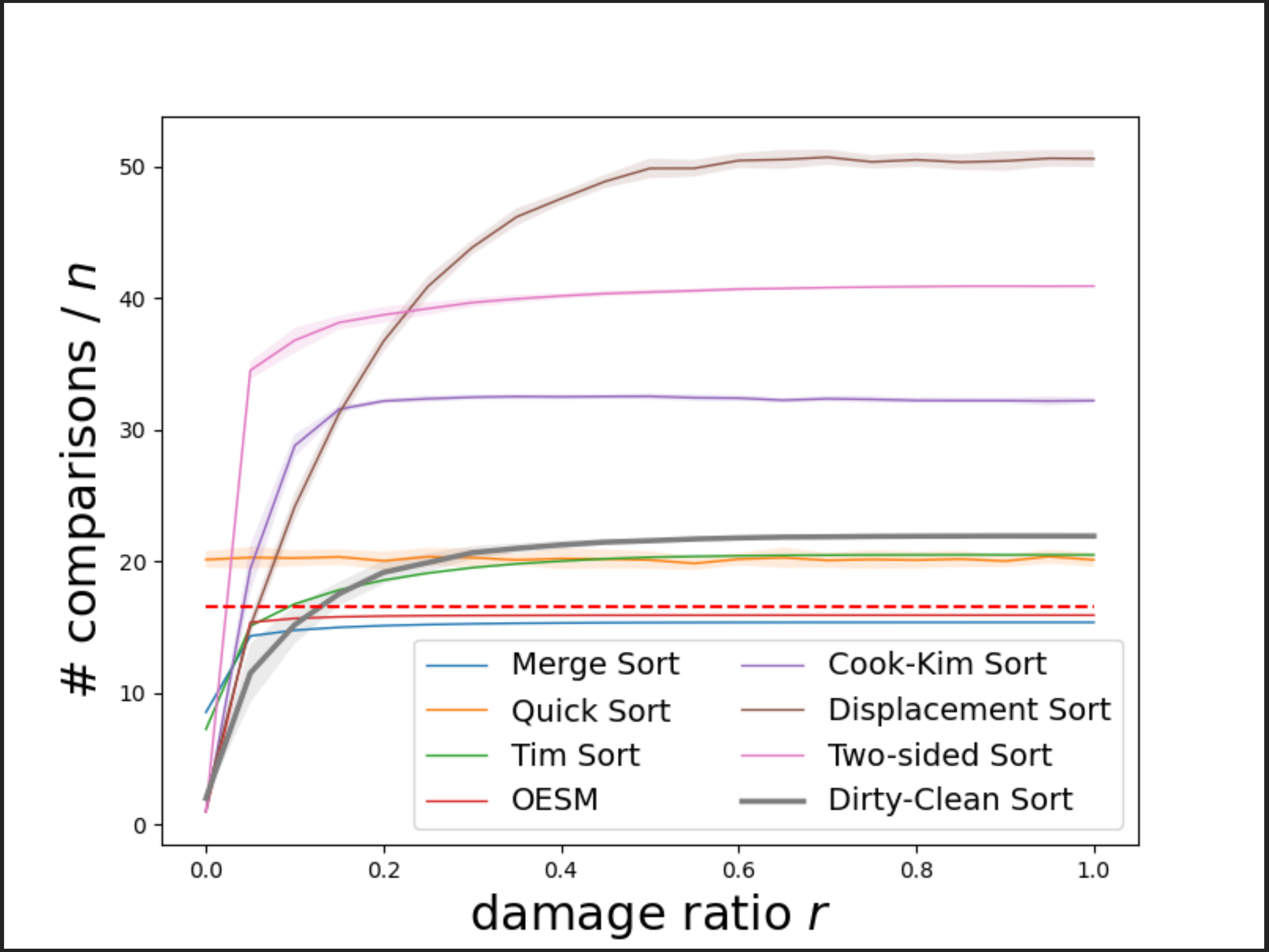
Repeatedly add ± 1 to $\hat{p}(i)$, for i random (n=1,000,000)



Experiments

Fraction r of items damaged (n=100,000)

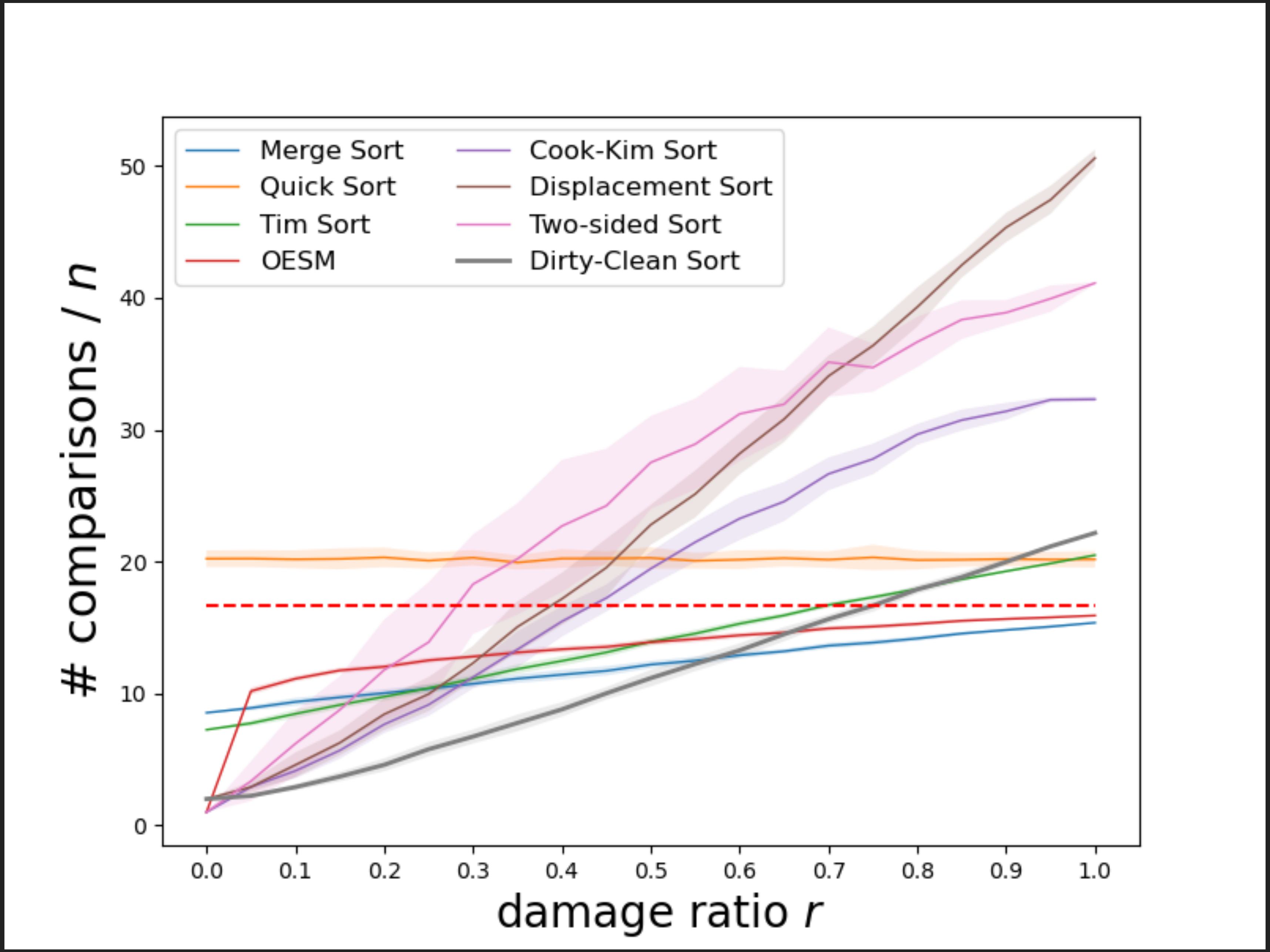
$\hat{<}$ random if an item damaged, otherwise correct



Experiments

Fraction r of items damaged (n=100,000)

$\hat{<}$ random if both items damaged, otherwise correct



Roadmap

- Sorting with Predictions
- Weighted Paging with Predictions (today + tomorrow)
- Mixing Multiple Predictions (tomorrow or Thursday)
- Shortest Paths without a Map (Thursday)
- Randomized k-Server Conjecture (Thursday)

Learning-Augmented Weighted Paging

JOINT WORK WITH

NIKHIL BANSAL (UNIVERSITY OF MICHIGAN)
RAVI KUMAR (GOOGLE)
MANISH PUROHIT (GOOGLE)
ERIK VEE (GOOGLE)

The paging problem (aka caching)

- cache can hold k pages

The paging problem (aka caching)

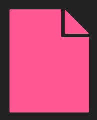
- cache can hold k pages

- at time $t = 1,2,\ldots$
  - page p_t requested

The paging problem (aka caching)

- cache can hold k pages

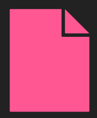
- at time $t = 1, 2, \ldots$
  - page p_t requested

Request: 

The paging problem (aka caching)

- cache can hold k pages

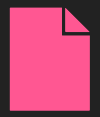
- at time $t = 1, 2, \ldots$

Request: 

  - page p_t requested
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache
    - pay 1

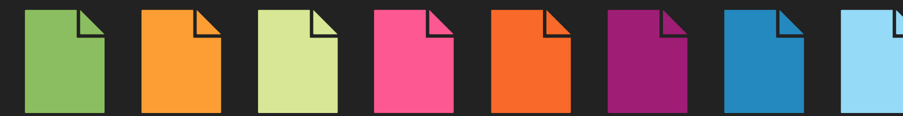
The paging problem (aka caching)

- cache can hold k pages
- at time $t = 1, 2, \ldots$
  - page p_t requested
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache
    - pay 1

Request: 

The paging problem (aka caching)

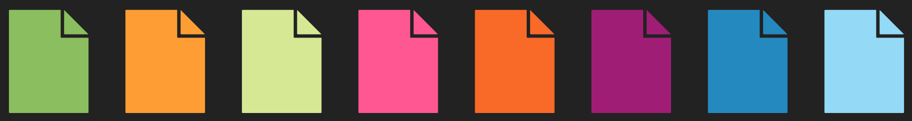
- cache can hold k pages
- at time $t = 1, 2, \ldots$
  - page p_t requested
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache
    - pay 1

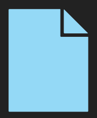


Request:

The paging problem (aka caching)

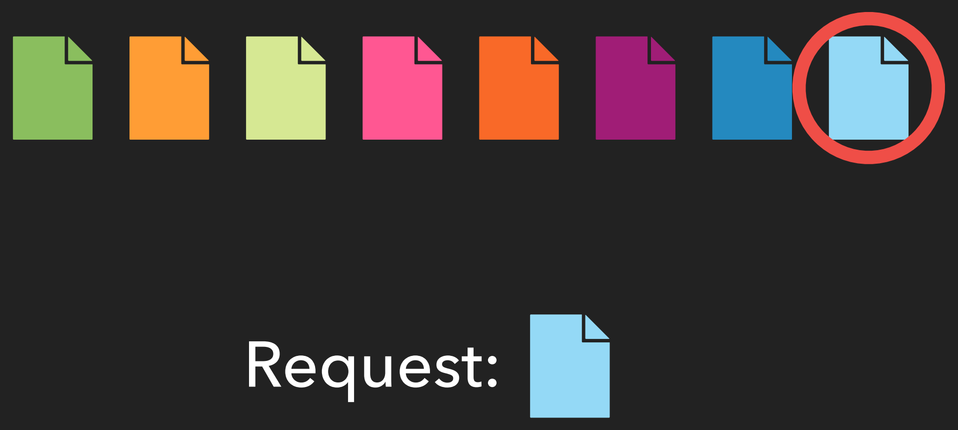
- cache can hold k pages
- at time $t = 1,2,\ldots$
  - page p_t requested
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache
    - pay 1



Request: 

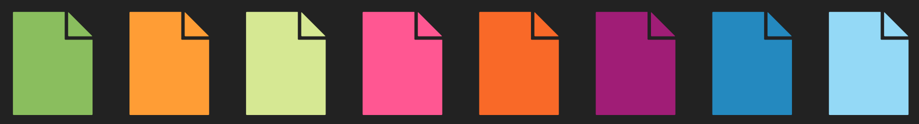
The paging problem (aka caching)

- cache can hold k pages
- at time $t = 1,2,\ldots$
  - page p_t requested
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache
    - pay 1



The paging problem (aka caching)

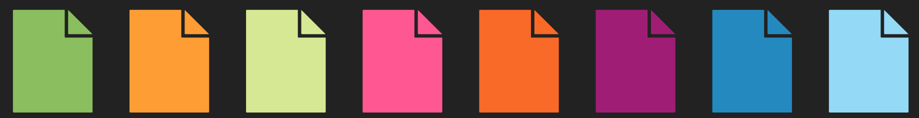
- cache can hold k pages
- at time $t = 1,2,\ldots$
  - page p_t requested
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache
    - pay 1



Request:

The paging problem (aka caching)

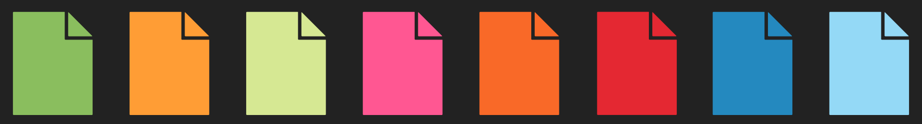
- cache can hold k pages
- at time $t = 1, 2, \ldots$
  - page p_t requested
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache
    - pay 1



Request: 

The paging problem (aka caching)

- cache can hold k pages
- at time $t = 1,2,\ldots$
  - page p_t requested
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache
    - pay 1



Request:

Algorithm FIF [Belady 66]

On cache miss, evict page whose next request is farthest in future.

Algorithm FIF [Belady 66]

On cache miss, evict page whose next request is farthest in future.

Theorem: FIF is optimal.

Algorithm FIF [Belady 66]

On cache miss, evict page whose next request is farthest in future.

Theorem: FIF is optimal.

Proof: Let

$n(s)$:= #pages in FIF-cache in pos $\leq s$ in next-request ordering

$n^*(s)$:= ––––––– OPT-cache ––––––––––––––––––––––––––

Algorithm FIF [Belady 66]

On cache miss, evict page whose next request is farthest in future.

Theorem: FIF is optimal.

Proof: Let

$n(s)$:= #pages in FIF-cache in pos $\leq s$ in next-request ordering

$n^*(s)$:= ––––––– OPT-cache ––––––––––––––––––––––––––––––

$$\Phi := \max_s n^*(s) - n(s)$$

Algorithm FIF [Belady 66]

On cache miss, evict page whose next request is farthest in future.

Theorem: FIF is optimal.

Proof: Let

$n(s)$:= #pages in FIF-cache in pos $\leq s$ in next-request ordering

$n^*(s)$:= ––––––– OPT-cache ––––––––––––––––––––––––––

$$\Phi := \max_s n^*(s) - n(s)$$

Claim: $\Delta\text{cost}_{\text{FIF}} + \Delta\Phi \leq \Delta\text{cost}_{\text{OPT}}$

Algorithm FIF [Belady 66]

On cache miss, evict page whose next request is farthest in future.

Theorem: FIF is optimal.

Proof: Let

$n(s)$:= #pages in FIF-cache in pos $\leq s$ in next-request ordering

$n^*(s)$:= ––––––– OPT-cache ––––––––––––––––––––––––––––

$$\Phi := \max_s n^*(s) - n(s)$$

Claim: $\Delta\text{cost}_{\text{FIF}} + \Delta\Phi \leq \Delta\text{cost}_{\text{OPT}}$ $\square$

Algorithm FIF [Belady 66]

On cache miss, evict page whose next request is farthest in future.

Theorem: FIF is optimal.

Online paging

Requests revealed one by one

Algorithm ρ-competitive if

$$\text{cost} \leq \rho \cdot \text{opt} + \text{const}$$

Algorithm FIF [Belady 66]

On cache miss, evict page whose next request is farthest in future.

Theorem: FIF is optimal.

Online paging

Requests revealed one by one

Algorithm ρ-competitive if

$$\text{cost} \leq \rho \cdot \text{opt} + \text{const}$$

Competitive ratio of paging: k deterministically [ST85]

$\Theta(\log k)$ randomized [FKLMSY91]

Learning-Augmented Paging [Lykouris & Vassilvitskii 18]

At time t, page p_t requested and additional input:

$\tau_t :=$ predicted time when p_t next requested

Learning-Augmented Paging [Lykouris & Vassilvitskii 18]

At time t, page p_t requested and additional input:

$\tau_t :=$ predicted time when p_t next requested

[LV 18]: $O\left(\min\left\{\sqrt{\frac{\eta}{\text{opt}}}, \log k\right\}\right)$-competitive, where

$$\eta := \sum_t |\underbrace{\tau_t}_{\text{prediction}} - \underbrace{a_t}_{\text{truth}}|$$

Learning-Augmented Paging [Lykouris & Vassilvitskii 18]

At time t, page p_t requested and additional input:

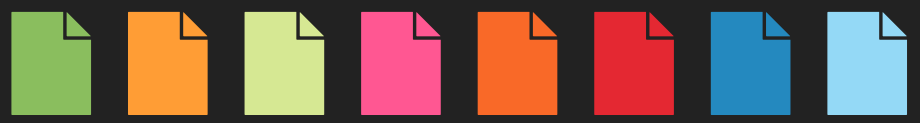
$$\tau_t := \text{predicted time when } p_t \text{ next requested}$$

[LV 18]: $O\left(\min\left\{\sqrt{\frac{\eta}{\mathrm{opt}}}, \log k\right\}\right)$-competitive, where

$$\eta := \sum_t |\underbrace{\tau_t}_{\text{prediction}} - \underbrace{a_t}_{\text{truth}}|$$

[Rohatgi 20, Wei 20]: Improved dependence on η

Weighted Paging



- cache can hold k pages
- Page p has weight $w_p > 0$
- at time $t = 1,2,\ldots$
  - page p_t requested
  - if $p_t \notin$ cache (“cache miss”)
    - evict a page from cache
    - fetch p_t into cache
    - pay w_{p_t}

Weighted Paging

- Stepping stone towards k-server

- Same comp. ratio as paging (k determ., $\Theta(\log k)$ rand.), but techniques more challenging

Weighted Paging

- Stepping stone towards k-server
- Same comp. ratio as paging (k determ., $\Theta(\log k)$ rand.), but techniques more challenging

FIF with weights?

Weighted Paging

- Stepping stone towards k-server
- Same comp. ratio as paging (k determ., $\Theta(\log k)$ rand.), but techniques more challenging

FIF with weights?

Weighted paging with predictions?

Bad News

$\Omega(k)$ deterministic and $\Omega(\log k)$ randomized even with accurate predictions of next request times [JPS20,ACEPS20]

Bad News

$\Omega(k)$ deterministic and $\Omega(\log k)$ randomized even with accurate predictions of next request times [JPS20,ACEPS20]

[JPS20]: Prediction of *entire request sequence* until each page requested again

[ACEPS20]: Prediction of optimal actions

Learning-Augmented Weighted Paging

$\Omega(l)$ deterministic and $\Omega(\log l)$ randomized with accurate predictions of next request times [JPS20,ACEPS20]

where $l =$ #weight classes

Learning-Augmented Weighted Paging

$\Omega(l)$ deterministic and $\Omega(\log l)$ randomized with accurate predictions of next request times [JPS20,ACEPS20]

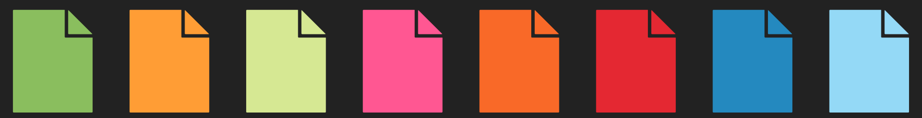
where $l =$ #weight classes

[Bansal,Coester,Kumar,Purohit,Vee 22]:

Theorem: These bounds are tight.

Recap: Learning-Augmented Weighted Paging

- cache can hold k pages

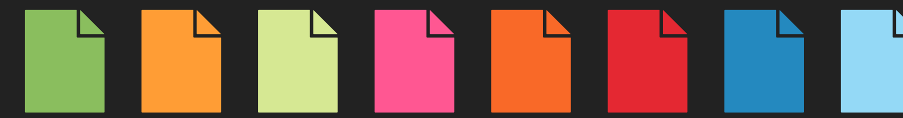


- at time $t = 1,2,\ldots$
  - page p_t requested
  - prediction τ_t of next time when p_t requested again
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache for cost w_{p_t}

Parameter: $l =$ #weight classes

Recap: Learning-Augmented Weighted Paging

- cache can hold k pages



- at time $t = 1,2,\ldots$
  - page p_t requested
  - prediction τ_t of next time when p_t requested again
  - if $p_t \notin$ cache ("cache miss")
    - evict a page from cache
    - fetch p_t into cache for cost w_{p_t}

Parameter: $l =$ #weight classes $\leq O\left(\log \frac{\max_p w_p}{\min_p w_p}\right)$

Theorem: [Bansal,Coester,Kumar,Purohit,Vee 22]

If predictions accurate, there is

- l-competitive deterministic algorithm
- $O(\log l)$-competitive randomized algorithm

Theorem: [Bansal,Coester,Kumar,Purohit,Vee 22]

If predictions unreliable, there is

- $O\left(\min\left\{l + \frac{l \cdot \epsilon}{\text{opt}}, k\right\}\right)$-competitive deterministic algorithm
- $O\left(\min\left\{\log l + \frac{l \cdot \epsilon}{\text{opt}}, \log k\right\}\right)$-competitive randomized algorithm

where $\epsilon = \sum_{i=1}^{l} w_i \cdot$ #surprises in weight class i

Theorem: [Bansal,Coester,Kumar,Purohit,Vee 22]

If predictions unreliable, there is

- $O\left(\min\left\{l + \frac{l \cdot \epsilon}{\text{opt}}, k\right\}\right)$-competitive deterministic algorithm
- $O\left(\min\left\{\log l + \frac{l \cdot \epsilon}{\text{opt}}, \log k\right\}\right)$-competitive randomized algorithm

where $\epsilon = \sum_{i=1}^{l} w_i \cdot \text{\#surprises in weight class i}$

$$\leq 2 \sum_t w_{p_t} \cdot |\tau_t - a_t|$$

Assume now accurate predictions

Assume now accurate predictions

(Extension to inaccurate predictions fairly simple)

Algorithm consists of global strategy and local strategy

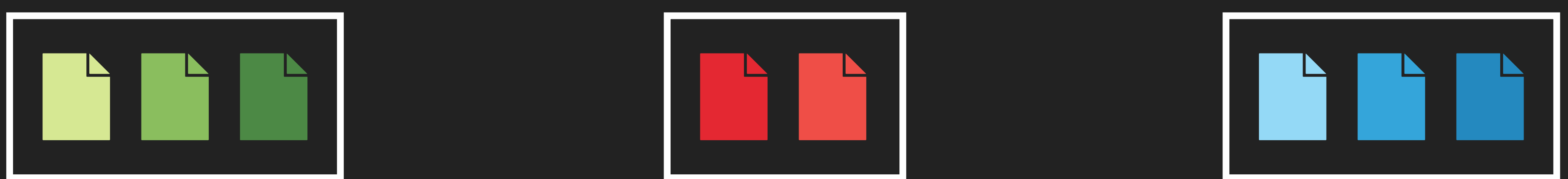
How many pages x_i from each weight class i?

Which x_i pages?

Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?



$x_1 = 3$ $x_2 = 2$ $x_3 = 3$

$k = 8$

$l = 3$

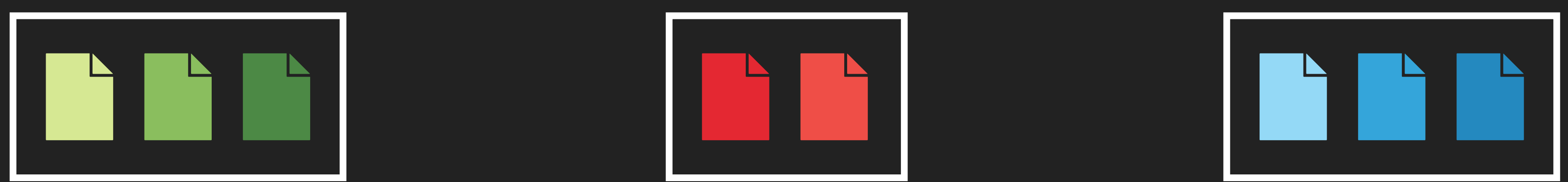
Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?

$k = 8$

$l = 3$

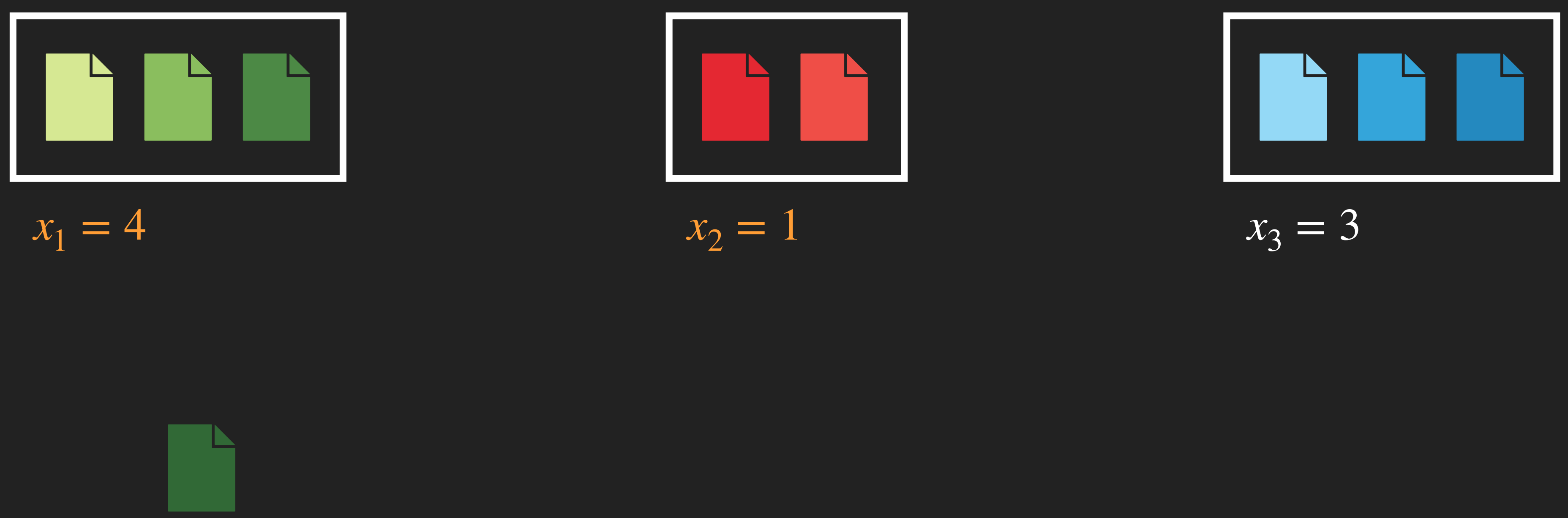


$x_1 = 3$ $x_2 = 2$ $x_3 = 3$

Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?



$k = 8$

$l = 3$

Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?

$x_1 = 4$ $x_2 = 1$ $x_3 = 3$

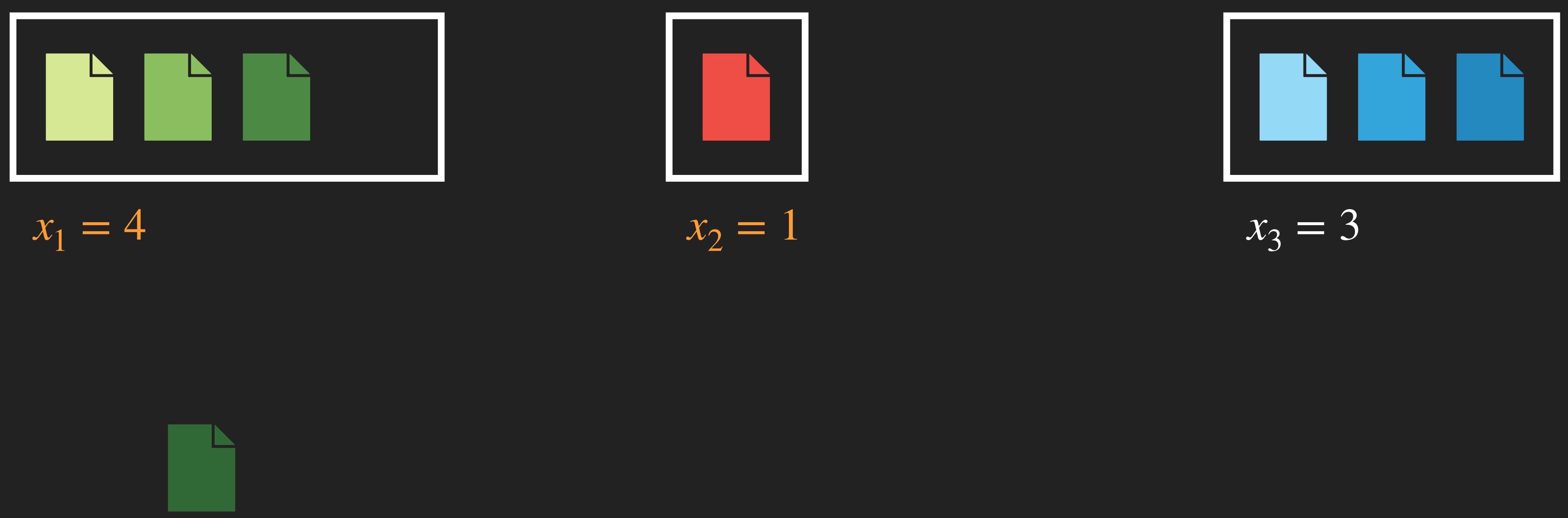
$k = 8$

$l = 3$

Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?



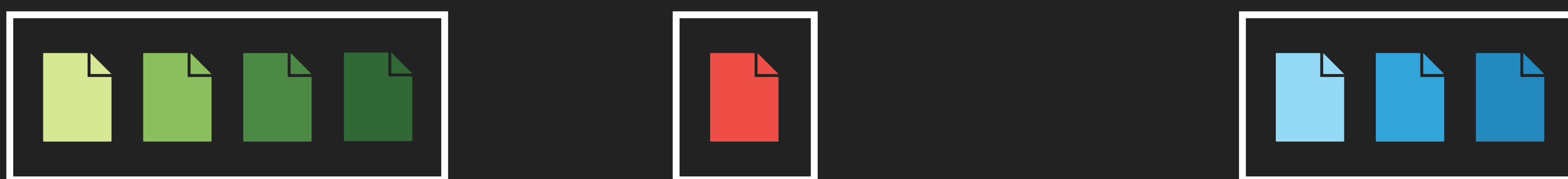
$k = 8$

$l = 3$

Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?



$x_1 = 4$ $x_2 = 1$ $x_3 = 3$

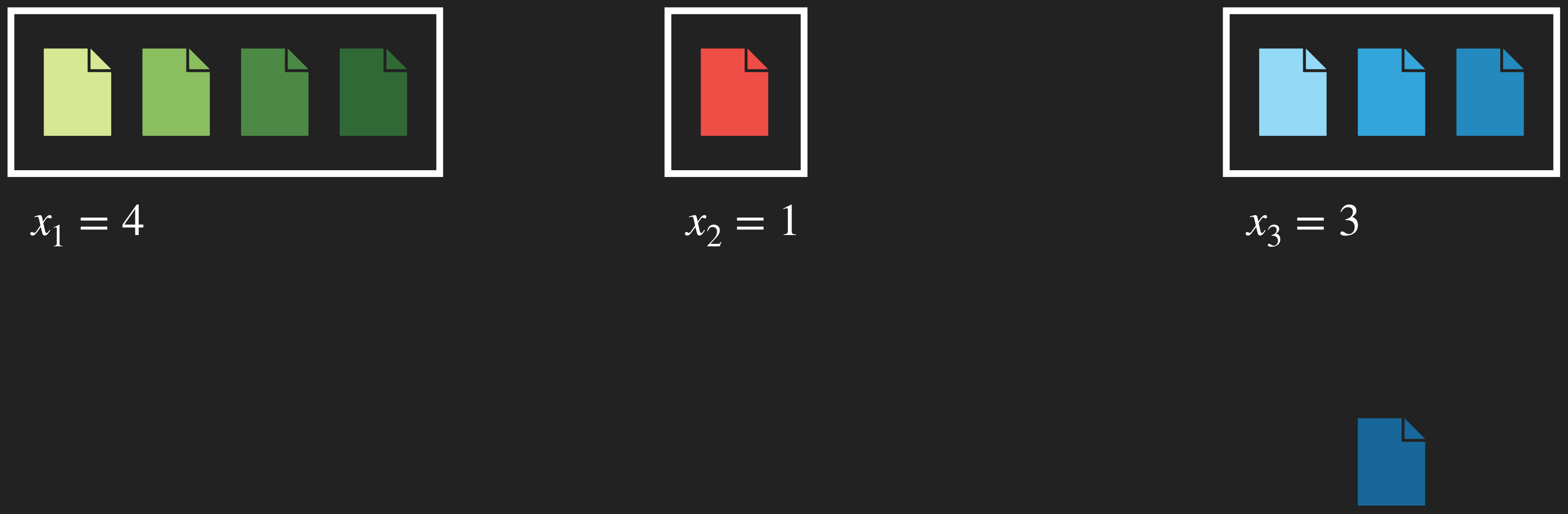
$k = 8$

$l = 3$

Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?



$x_1 = 4$ $x_2 = 1$ $x_3 = 3$

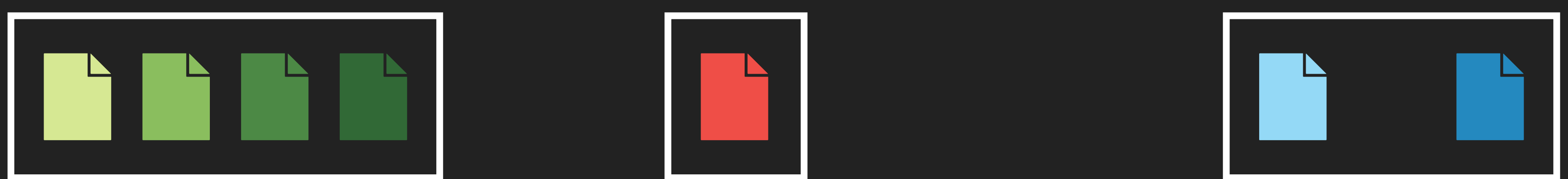
$k = 8$

$l = 3$

Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?



$x_1 = 4$ $\quad$ $x_2 = 1$ $\quad$ $x_3 = 3$

$k = 8$

$l = 3$

Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?

$x_1 = 4$ $x_2 = 1$ $x_3 = 3$

$k = 8$

$l = 3$

Algorithm consists of global strategy and local strategy

How many pages x_i from each weight class i?

Which x_i pages?

Deterministic Algorithm:

$x_i := \lceil \tilde{x}_i \rceil$ for some $\tilde{x}_i \geq 0$

Global: $\tilde{x}_i' = -\frac{1}{w_i}$ until $\sum_i \lceil \tilde{x}_i \rceil \leq k - 1$

Then $\tilde{x}_r := \tilde{x}_r + 1$, where $r =$ class of requested page

Local: FIF

$O(\log l)$-competitive randomized algorithm

Page ranks

Consider sub-instance of single weight class

Let C_t^m = pages in cache of FIF at time t if cache size m

Page ranks

Consider sub-instance of single weight class

Let C_t^m = pages in cache of FIF at time t if cache size m

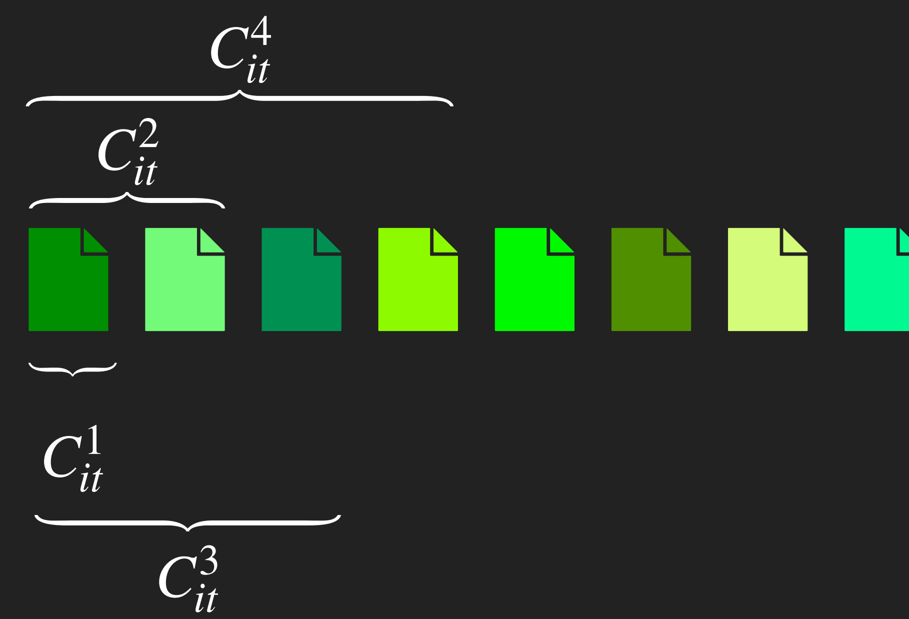
Lemma: $C_t^1 \subset C_t^2 \subset C_t^3 \subset \ldots \forall t$

Page ranks

Consider sub-instance of single weight class

Let C_t^m = pages in cache of FIF at time t if cache size m

Lemma: $C_t^1 \subset C_t^2 \subset C_t^3 \subset \ldots \forall t$



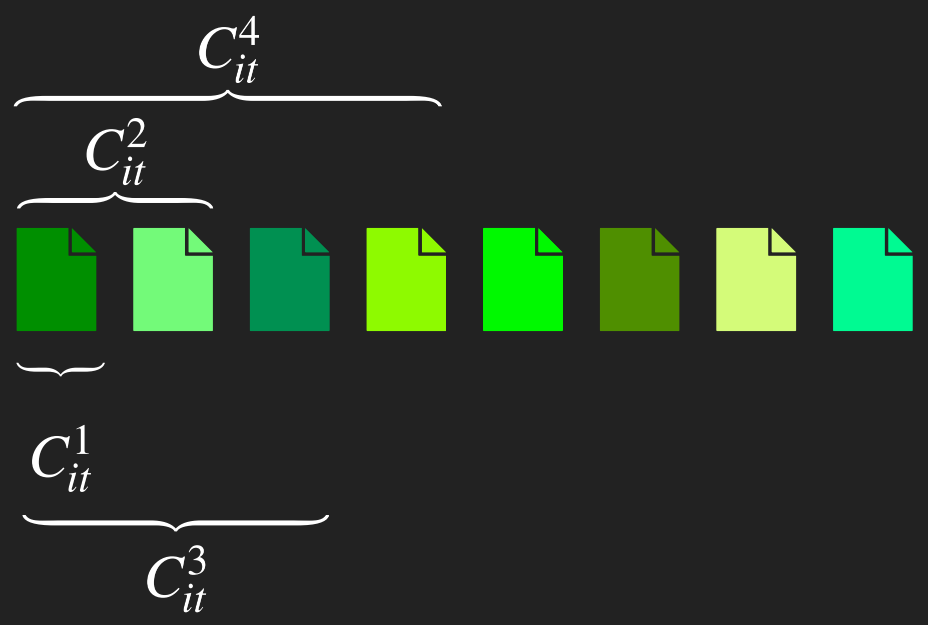
Page ranks

Consider sub-instance of single weight class

Let C_t^m = pages in cache of FIF at time t if cache size m

Lemma: $C_t^1 \subset C_t^2 \subset C_t^3 \subset \ldots \forall t$

Proof: Induction on t. $\square$



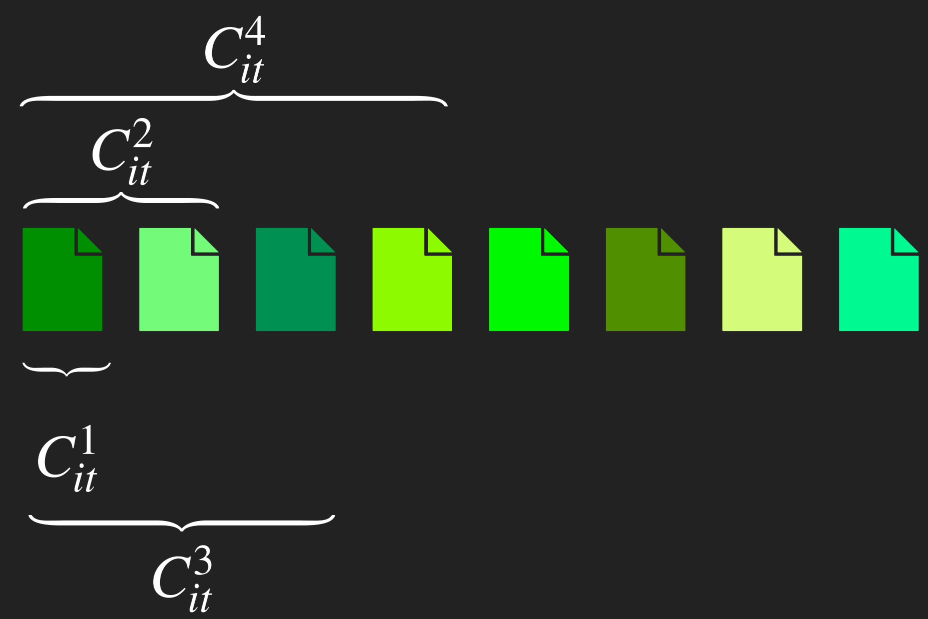
Page ranks

Consider sub-instance of single weight class

Let C_t^m = pages in cache of FIF at time t if cache size m

Lemma: $C_t^1 \subset C_t^2 \subset C_t^3 \subset \ldots \forall t$

Proof: Induction on t. $\square$



Definition: Page p has rank m at time t if $C_t^m \setminus C_t^{m-1} = \{p\}$

Change of ranks

Lemma: Let $p_1, p_2, \ldots$ be pages sorted by rank at time t.

Change of ranks

Lemma: Let $p_1, p_2, \ldots$ be pages sorted by rank at time t.

Let p_{m_0} be requested next.

Change of ranks

Lemma: Let $p_1, p_2, \ldots$ be pages sorted by rank at time t.

Let p_{m_0} be requested next.

Let $m_0 > m_1 > m_2 > \ldots$ s.t.

$$p_{m_{i+1}} \text{ is farthest-in-future among } C_t^{m_i - 1} = \{p_1, p_2, \ldots, p_{m_i - 1}\}$$

Change of ranks

Lemma: Let $p_1, p_2, \ldots$ be pages sorted by rank at time t.

Let p_{m_0} be requested next.

Let $m_0 > m_1 > m_2 > \ldots$ s.t.

$p_{m_{i+1}}$ is farthest-in-future among $C_t^{m_i - 1} = \{p_1, p_2, \ldots, p_{m_i - 1}\}$

Then at time $t + 1$:

- p_{m_0} has rank 1
- p_{m_i} has rank m_{i-1}
- other ranks unchanged

Local strategy

From class i, have pages with ranks $\leq \lfloor x_i \rfloor$ fully in cache,

page with rank $\lceil x_i \rceil$ fractionally.

Local strategy

From class i, have pages with ranks $\leq \lfloor x_i \rfloor$ fully in cache,

page with rank $\lceil x_i \rceil$ fractionally.

Concisely: cache $= \bigcup_{i=1}^{l} C_{it}^{x_i}$

Local strategy

From class i, have pages with ranks $\leq \lfloor x_i \rfloor$ fully in cache,

page with rank $\lceil x_i \rceil$ fractionally.

Concisely: cache $= \bigcup_{i=1}^{l} C_{it}^{x_i}$

Lemma: This is best local strategy, up to factor 3.

Local strategy

From class i, have pages with ranks $\leq \lfloor x_i \rfloor$ fully in cache,

page with rank $\lceil x_i \rceil$ fractionally.

Concisely: cache $= \displaystyle\bigcup_{i=1}^{l} C_{it}^{x_i}$

Lemma: This is best local strategy, up to factor 3.

Proof idea: Use potential

$$\Phi = \sum_{i=1}^{l} w_i \cdot \max_s \left(n_i^*(s) - n_i(s) \right) \qquad \text{where}$$

$n_i(s)$:= #pages in cache in pos $\leq s$ in next-request ordering of class-i-subinstance

$n_i^*(s)$:= ..... in optimal cache ...... $\square$

$$\text{online cache} = \bigcup_{i=1}^{l} C_{it}^{x_i(t)}$$

$$\text{offline cache} = \bigcup_{i=1}^{l} C_{it}^{y_i(t)} \text{ for unknown } y_i(t)$$

$$\text{online cache} = \bigcup_{i=1}^{l} C_{it}^{x_i(t)}$$

$$\text{offline cache} = \bigcup_{i=1}^{l} C_{it}^{y_i(t)} \text{ for unknown } y_i(t)$$

When rank-r-page of class i requested:

online cache miss iff $r > x_i$

offline cache miss iff $r > y_i$

Global strategy problem: Geometric view

$$\Delta := \left\{ x \in \mathbb{R}_+^l \;\middle|\; \sum_i x_i = k \right\}$$

At time $t = 1,2,\ldots$

- $(i_t, r_t) \in [l] \times \mathbb{N}$ revealed
- Algo chooses $x(t) \in \Delta$
- Pay $w_{i_t} \cdot c_{r_t}(x_{i_t}) + \sum_{i\in[l]} w_i \,|\, x_i(t) - x_i(t-1) \,|$

where $c_r(z) := \begin{cases} 1 & z \leq r-1 \\ 0 & z \geq r \\ \text{linear} & z \in [r-1, r] \end{cases}$

Global strategy problem: Geometric view

$$\Delta := \left\{ x \in \mathbb{R}^l_+ \;\middle|\; \sum_i x_i = k \right\}$$

At time $t = 1,2,\ldots$

- $(i_t, r_t) \in [l] \times \mathbb{N}$ revealed
- Algo chooses $x(t) \in \Delta$
- Pay $w_{i_t} \cdot c_{r_t}(x_{i_t}) + \sum_{i \in [l]} w_i \,|\, x_i(t) - x_i(t-1) \,|$

where $c_r(z) := \begin{cases} 1 & z \leq r-1 \\ 0 & z \geq r \\ \text{linear} & z \in [r-1, r] \end{cases}$

HOWEVER: This problem has $\Omega(l)$ lower bound!

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2,3,\ldots,r-1$ are requested.

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2, 3, \ldots, r-1$ are requested.

$2\ 3\ 4\ 5\ 6\ 2\ 3\ 2\ 4\ 5\ 3\ 6$ possible

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2, 3, \ldots, r-1$ are requested.

$2\ 3\ 4\ 5\ 6\ 2\ 3\ 2\ 4\ 5\ 3\ 6$ possible

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2, 3, \ldots, r-1$ are requested.

$2\ 3\ 4\ 5\ 6\ 2\ 3\ 2\ 4\ 5\ 3\ 6$ possible

$2\ 3\ 4\ 5\ 6\ 2\ 5$

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2, 3, \ldots, r-1$ are requested.

$2\ 3\ 4\ 5\ 6\ 2\ 3\ 2\ 4\ 5\ 3\ 6$ possible

$2\ 3\ 4\ 5\ 6\ 2\ 5$

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2,3,\ldots,r-1$ are requested.

$2\ 3\ 4\ 5\ 6\ 2\ 3\ 2\ 4\ 5\ 3\ 6$ possible

$2\ 3\ 4\ 5\ 6\ 2\ 5$ impossible (unless predictions erroneous)

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2, 3, \ldots, r-1$ are requested.

Proof:

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2, 3, \ldots, r-1$ are requested.

Proof: Suppose r requested at times $t_1 < t_2$ and some $r' \in \{2, \ldots, r-1\}$ not requested in between.

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2, 3, \ldots, r-1$ are requested.

Proof: Suppose r requested at times $t_1 < t_2$ and some $r' \in \{2, \ldots, r-1\}$ not requested in between.

rank r page

Let $t \in [t_1, t_2 - 1]$ last time when identity of $p_r(t)$ changes

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2,3,\ldots,r-1$ are requested.

Proof: Suppose r requested at times $t_1 < t_2$ and some $r' \in \{2,\ldots,r-1\}$ not requested in between.

rank r page

Let $t \in [t_1, t_2 - 1]$ last time when identity of $p_r(t)$ changes

$\rightsquigarrow p_r(t)$ is farthest in future among $p_2(t), \ldots, p_r(t)$

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2,3,\ldots,r-1$ are requested.

Proof: Suppose r requested at times $t_1 < t_2$ and some $r' \in \{2,\ldots,r-1\}$ not requested in between.

rank r page

Let $t \in [t_1, t_2 - 1]$ last time when identity of $p_r(t)$ changes

$\rightsquigarrow p_r(t)$ is farthest in future among $p_2(t), \ldots, p_r(t)$

In particular: $p'_r(t)$ due to be requested earlier than $p_r(t)$

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2,3,\ldots,r-1$ are requested.

Proof: Suppose r requested at times $t_1 < t_2$ and some $r' \in \{2,\ldots,r-1\}$ not requested in between.

rank r page

Let $t \in [t_1, t_2 - 1]$ last time when identity of $p_r(t)$ changes

$\rightsquigarrow p_r(t)$ is farthest in future among $p_2(t), \ldots, p_r(t)$

In particular: $p'_r(t)$ due to be requested earlier than $p_r(t)$

This remains true until time t_2 – Contradiction.

Repeat property

Lemma: Between two requests to rank r of class i, all ranks $2,3,\ldots,r-1$ are requested.

Proof: Suppose r requested at times $t_1 < t_2$ and some $r' \in \{2,\ldots,r-1\}$ not requested in between.

rank r page

Let $t \in [t_1, t_2 - 1]$ last time when identity of $p_r(t)$ changes

$\rightsquigarrow p_r(t)$ is farthest in future among $p_2(t), \ldots, p_r(t)$

In particular: $p'_r(t)$ due to be requested earlier than $p_r(t)$

This remains true until time t_2 – Contradiction.

$\square$

Global strategy problem: Geometric view

$$\Delta := \left\{ x \in \mathbb{R}_+^l \;\middle|\; \sum_i x_i = k \right\}$$

At time $t = 1,2,\ldots$

- $(i_t, r_t) \in [l] \times \mathbb{N}$ arrives
- Algo chooses $x(t) \in \Delta$
- Pay $w_{i_t} \cdot c_{r_t}(x_{i_t}) + \sum_{i \in [l]} w_i \,|\, x_i(t) - x_i(t-1) \,|$

where $c_r(z) := \begin{cases} 1 & z \leq r-1 \\ 0 & z \geq r \\ \text{linear} & z \in [r-1, r] \end{cases}$

Global strategy problem: Geometric view

$$\Delta := \left\{ x \in \mathbb{R}^l_+ \;\middle|\; \sum_i x_i = k \right\}$$

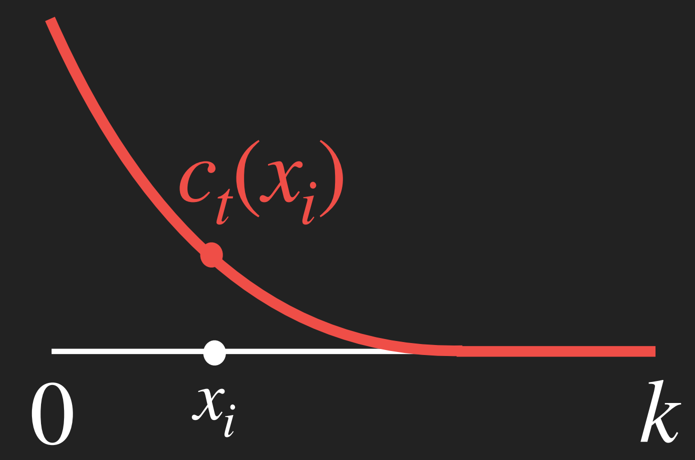
At time $t = 1,2,\ldots$

- $(i_t, r_t) \in [l] \times \mathbb{N}$ arrives s.t. all $(i_t,2), (i_t,3), \ldots, (i_t, r_t - 1)$ arrived since previous arrival of (i_t, r_t)
- Algo chooses $x(t) \in \Delta$
- Pay $w_{i_t} \cdot c_{r_t}(x_{i_t}) + \sum_{i\in[l]} w_i \,|\, x_i(t) - x_i(t-1) \,|$

where $c_r(z) := \begin{cases} 1 & z \le r-1 \\ 0 & z \ge r \\ \text{linear} & z \in [r-1, r] \end{cases}$

A similar problem

$$\Delta := \left\{ x \in \mathbb{R}^l_+ \;\middle|\; \sum_i x_i = k \right\}$$

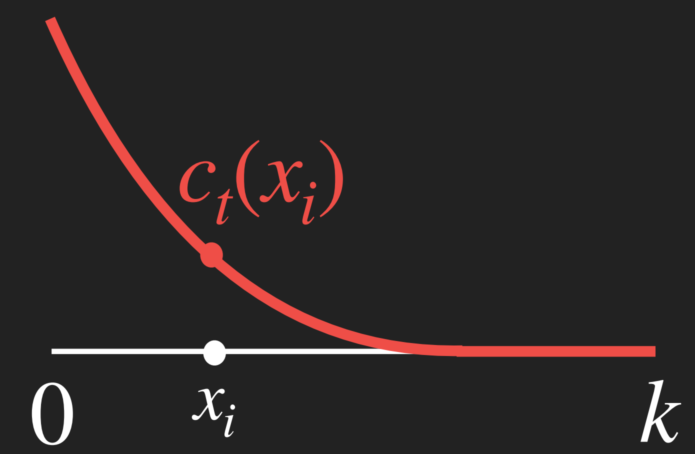


At time $t = 1,2,\ldots$

- Convex non-increasing $c_t\colon [0,k] \to \mathbb{R}_+$ arrives at $i_t \in [l]$
- Algo chooses $x(t) \in \Delta$
- Pay $w_{i_t} \cdot c_t(x_{i_t}) + \sum_{i \in [l]} w_i \,|\, x_i(t) - x_i(t-1) \,|$

A similar problem

$$\Delta := \left\{ x \in \mathbb{R}^l_+ \;\middle|\; \sum_i x_i = k \right\}$$



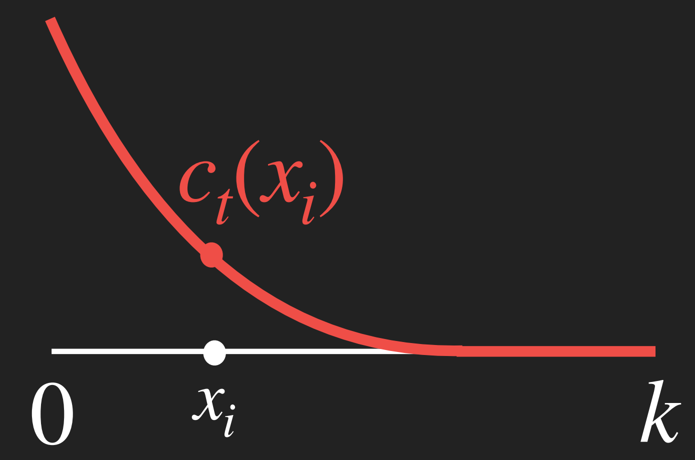
At time $t = 1,2,\ldots$

- Convex non-increasing $c_t \colon [0,k] \to \mathbb{R}_+$ arrives at $i_t \in [l]$
- Algo chooses $x(t) \in \Delta$
- Pay $w_{i_t} \cdot c_t(x_{i_t}) + \sum_{i\in[l]} w_i \,|\, x_i(t) - x_i(t-1) \,|$

This problem is $\Theta(\log l)$-competitive! [Bansal,Coester 22]

A similar problem

$$\Delta := \left\{ x \in \mathbb{R}^l_+ \;\middle|\; \sum_i x_i = k \right\}$$



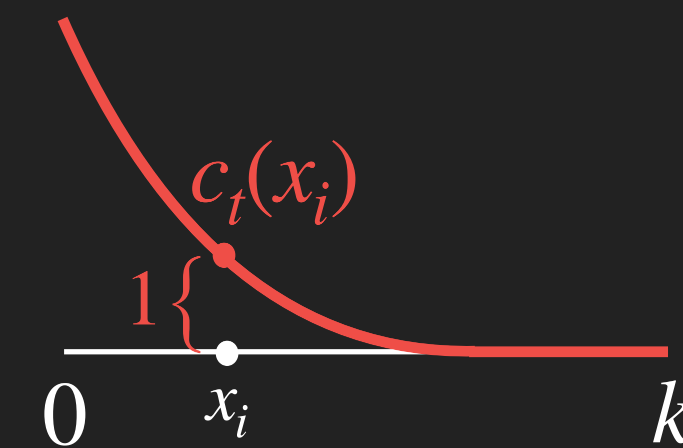
At time $t \in [0,\infty)$

- Convex non-increasing $c_t\colon [0,k] \to \mathbb{R}_+$ arrives at $i_t \in [l]$
- Algo chooses $x(t) \in \Delta$
- Pay $\displaystyle\int_0^\infty \left(w_{i_t} \cdot c_t(x_{i_t}) + \sum_{i \in [l]} w_i \,|\, x_i'(t) \,| \right) dt$

This problem is $\Theta(\log l)$-competitive! [Bansal,Coester 22]

A similar problem

$$\Delta := \left\{ x \in \mathbb{R}_+^l \;\middle|\; \sum_i x_i = k \right\}$$

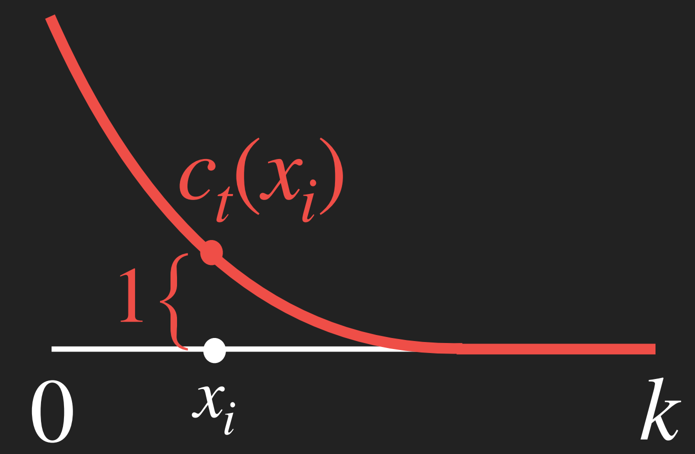


At time $t \in [0,\infty)$

- Convex non-increasing $c_t\colon [0,k] \to \mathbb{R}_+$ arrives at $i_t \in [l]$
- Algo chooses $x(t) \in \Delta$
- Pay $\displaystyle\int_0^\infty \left(w_{i_t} + \sum_{i\in[l]} w_i \,|\, x_i'(t) \,|\, \right) dt$

This problem is $\Theta(\log l)$-competitive! [Bansal,Coester 22]

Algorithm [Bansal,Coester 22]

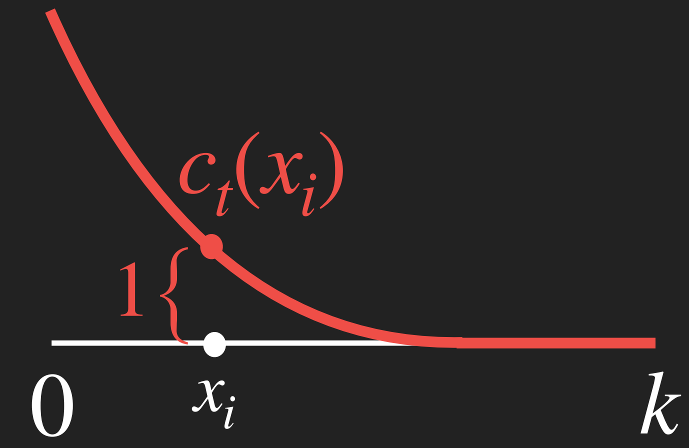


For each $i \in [l]$, auxiliary variable $\mu_i \geq 0$

At time $t \in [0,\infty)$

$$x_i'(t) = 1_{i=i_t} - \frac{\frac{\mu_i}{\sum_j \mu_j} + \frac{1}{l}}{B \cdot w_i}$$

Algorithm [Bansal,Coester 22]



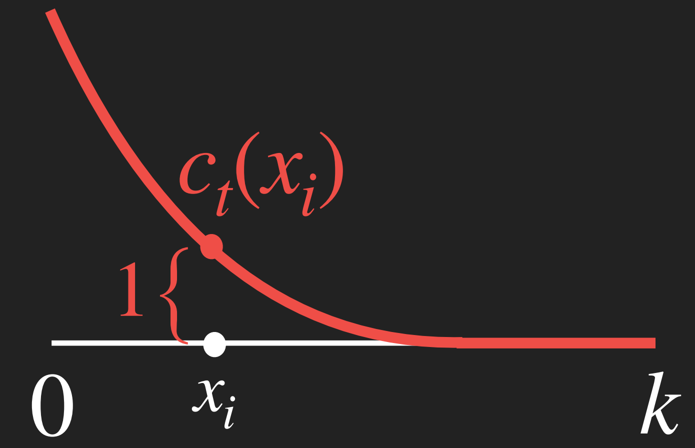
For each $i \in [l]$, auxiliary variable $\mu_i \geq 0$

willingness to decrease x_i

At time $t \in [0,\infty)$

$$x_i'(t) = 1_{i=i_t} - \frac{\frac{\mu_i}{\sum_j \mu_j} + \frac{1}{l}}{B \cdot w_i}$$

Algorithm [Bansal,Coester 22]



For each $i \in [l]$, auxiliary variable $\mu_i \geq 0$

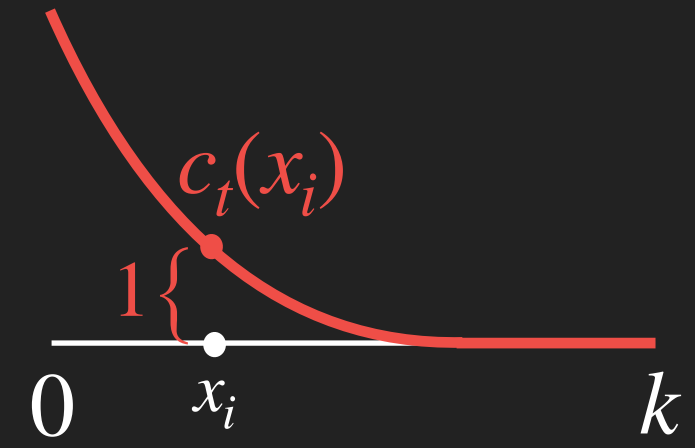
willingness to decrease x_i

At time $t \in [0,\infty)$

$$x_i'(t) = 1_{i=i_t} - \frac{\frac{\mu_i}{\sum_j \mu_j} + \frac{1}{l}}{B \cdot w_i}$$

$B > 0$ s.t. $\sum_i x_i'(t) = 0$

Algorithm [Bansal,Coester 22]



For each $i \in [l]$, auxiliary variable $\mu_i \geq 0$

willingness to decrease x_i

At time $t \in [0,\infty)$

$$x_i'(t) = 1_{i=i_t} - \frac{\frac{\mu_i}{\sum_j \mu_j} + \frac{1}{l}}{B \cdot w_i}$$

$B > 0$ s.t. $\sum_i x_i'(t) = 0$

$$\mu_i'(t) = \frac{\frac{\mu_i}{\sum_j \mu_j} + \frac{1}{l}}{B \cdot w_i} - 2 \cdot 1_{i=i_t \,\wedge\, w_i<\mu_i \cdot |c_t'(x_i)|}$$

Back to weighted paging

Cost function only amortized convex

Back to weighted paging

Cost function only amortized convex

- For $i \in [l]$, maintain set $S_i \subset [x_{i_t}, \infty)$

Idea: $S_i \approx$ recently requested ranks of class i

Back to weighted paging

Cost function only amortized convex

- For $i \in [l]$, maintain set $S_i \subset [x_{i_t}, \infty)$

Idea: $S_i \approx$ recently requested ranks of class i

- Set $\mu_i := |S_i|$ and update S_i such that

$$\mu_i'(t) = \frac{\frac{\mu_i}{\sum_j \mu_j} + \frac{1}{l}}{B \cdot w_i} - 2 \cdot 1_{i=i_t \,\wedge\, p \in S_i}$$

Back to weighted paging

Cost function only amortized convex

- For $i \in [l]$, maintain set $S_i \subset [x_{i_t}, \infty)$

Idea: $S_i \approx$ recently requested ranks of class i

- Set $\mu_i := |S_i|$ and update S_i such that

$$\mu_i'(t) = \frac{\frac{\mu_i}{\sum_j \mu_j} + \frac{1}{l}}{B \cdot w_i} - 2 \cdot 1_{i=i_t \wedge p \in S_i}$$

pointer that moves continuously through $[r_t - 1, r_t]$

Full algorithm (global strategy)

When rank r_t of class i_t requested:

- Move pointer p at rate 8 from $r_t - 1$ to r_t. Meanwhile:
  - If $p > x_{i_t}$
$$x_i'(t) = 1_{i=i_t} - \frac{\frac{\mu_i}{\sum_j \mu_j} + \frac{1}{l}}{B \cdot w_i}$$
  - $\forall i \neq i_t$: Add to S_i points passed by x_i
  - Shrink S_{i_t} at rate x_{i_t}' from left & rate 1 from right
  - If $p \notin S_{i_t}$: Grow S_{i_t} in $(r_t - 1, p]$ at rate 2

Analysis uses potential $10D + 2M + 5C + 4S$, where

$$D = \sum_i w_i \left(\mu_i + [x_i - y_i]_+\right) \log \frac{\left(1 + \frac{1}{l}\right)\left(\mu_i + [x_i - y_i]_+\right)}{\mu_i + \frac{1}{l}\left(\mu_i + [x_i - y_i]_+\right)}$$

$$M = \sum_i w_i \left[\mu_i - 2(y_i - x_i)\right]_+$$

$$C = \sum_i w_i \int_{S_i} \frac{|\,(y_i, x_i] \cap R_{iu}\,|}{\mu_i + \frac{1}{l}\left(\mu_i + |\,(y_i, x_i] \cap R_{iu}\,|\,\right)} du$$

$$S = \sum_i w_i \left(\,|\,S_i \cap [y_i, \infty)\,| + [x_i - y_i]_+\right)$$

Open problems

- Simpler algorithm/analysis?
- k-server with next-request time predictions?

Roadmap

- Sorting with Predictions
- Weighted Paging with Predictions
- Mixing Multiple Predictions
- Shortest Paths without a Map
- Randomized k-Server Conjecture

Mixing Predictions for Online Metric Algorithms

JOINT WORK WITH

ANTONIOS ANTONIADIS (UNIVERSITY OF TWENTE)
MAREK ELIAS (BOCCONI UNIVERSITY)
ADAM POLAK (MAX PLANCK INSTITUTE FOR INFORMATICS)
BERTRAND SIMON (IN2P3 COMPUTING CENTER / CNRS)

Metrical Task Systems (MTS) [Borodin, Linial, Saks 1987]

- metric space (M, d)
- At time $t = 1,2,\ldots$
  - $c_t \colon M \to \mathbb{R}_+ \cup \{\infty\}$ revealed
  - Choose $p_t \in M$
  - Pay $d(p_{t-1}, p_t) + c_t(p_t)$

Metrical Task Systems (MTS) [Borodin, Linial, Saks 1987]

- metric space (M, d)
- At time $t = 1,2,\ldots$
  - $c_t : M \to \mathbb{R}_+ \cup \{\infty\}$ revealed
  - Choose $p_t \in M$
  - Pay $d(p_{t-1}, p_t) + c_t(p_t)$

Examples:

paging, k-server, dynamic power management, convex body/function chasing, self-adjusting BSTs, …

Metrical Task Systems (MTS) with multiple predictors

- metric space (M, d)
- At time $t = 1,2,\ldots$
  - $c_t\colon M \to \mathbb{R}_+ \cup \{\infty\}$ revealed
  - Suggestions $\phi_{1t}, \phi_{2t}, \ldots, \phi_{kt} \in M$
  - Choose $p_t \in M$
  - Pay $d(p_{t-1}, p_t) + c_t(p_t)$

Metrical Task Systems (MTS) with multiple predictors

- metric space (M, d)
- At time $t = 1,2,\ldots$
  - $c_t \colon M \to \mathbb{R}_+ \cup \{\infty\}$ revealed
  - Suggestions $\phi_{1t}, \phi_{2t}, \ldots, \phi_{kt} \in M$
  - Choose $p_t \in M$
  - Pay $d(p_{t-1}, p_t) + c_t(p_t)$

Algo A is ρ-competitive against B if

$$\mathrm{cost}_A \leq \rho \cdot \mathrm{cost}_B + \mathrm{const}$$

Theorem: Against best dynamic combination, can be $\Theta(k^2)$ -competitive.

Theorem: Against best dynamic combination, can be $\Theta(k^2)$-competitive.

Theorem: Against best dynamic combination with limited switches, can be $(1+\epsilon)$-competitive.

Theorem: Against best dynamic combination, can be $\Theta(k^2)$-competitive.

Theorem: Against best dynamic combination with limited switches, can be $(1+\epsilon)$-competitive, even if only one suggestion queried per time step.

Theorem: Against best dynamic combination, can be $\Theta(k^2)$ -competitive.

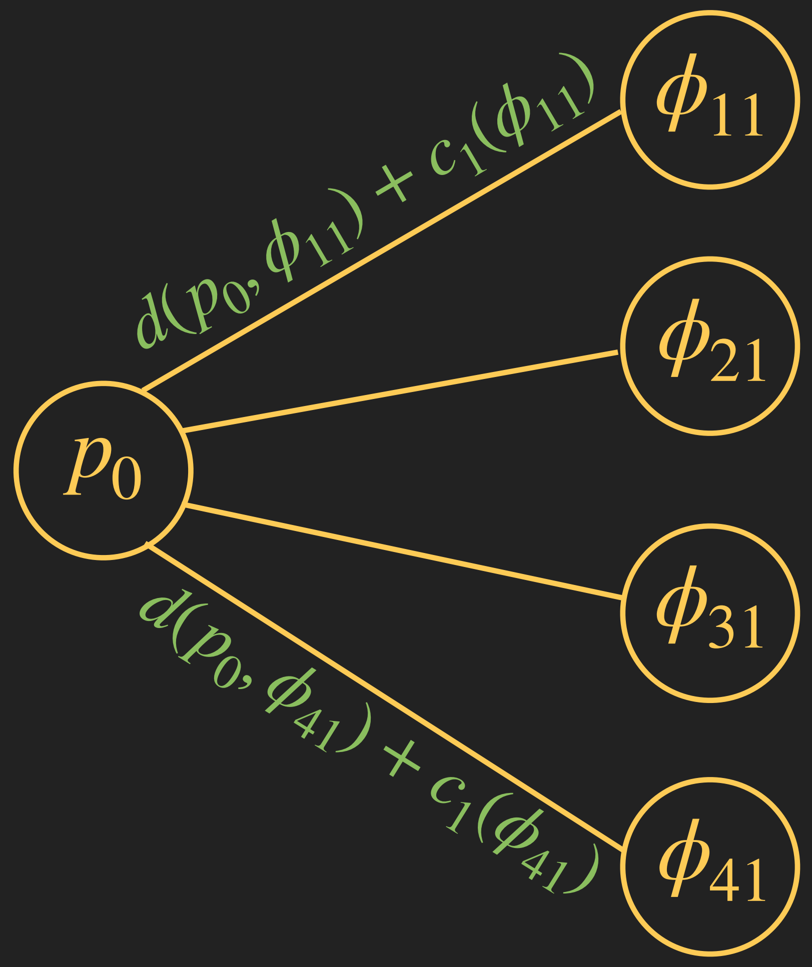
Theorem: Against best dynamic combination with limited switches, can be $(1+\epsilon)$-competitive, even if only one suggestion queried per time step.

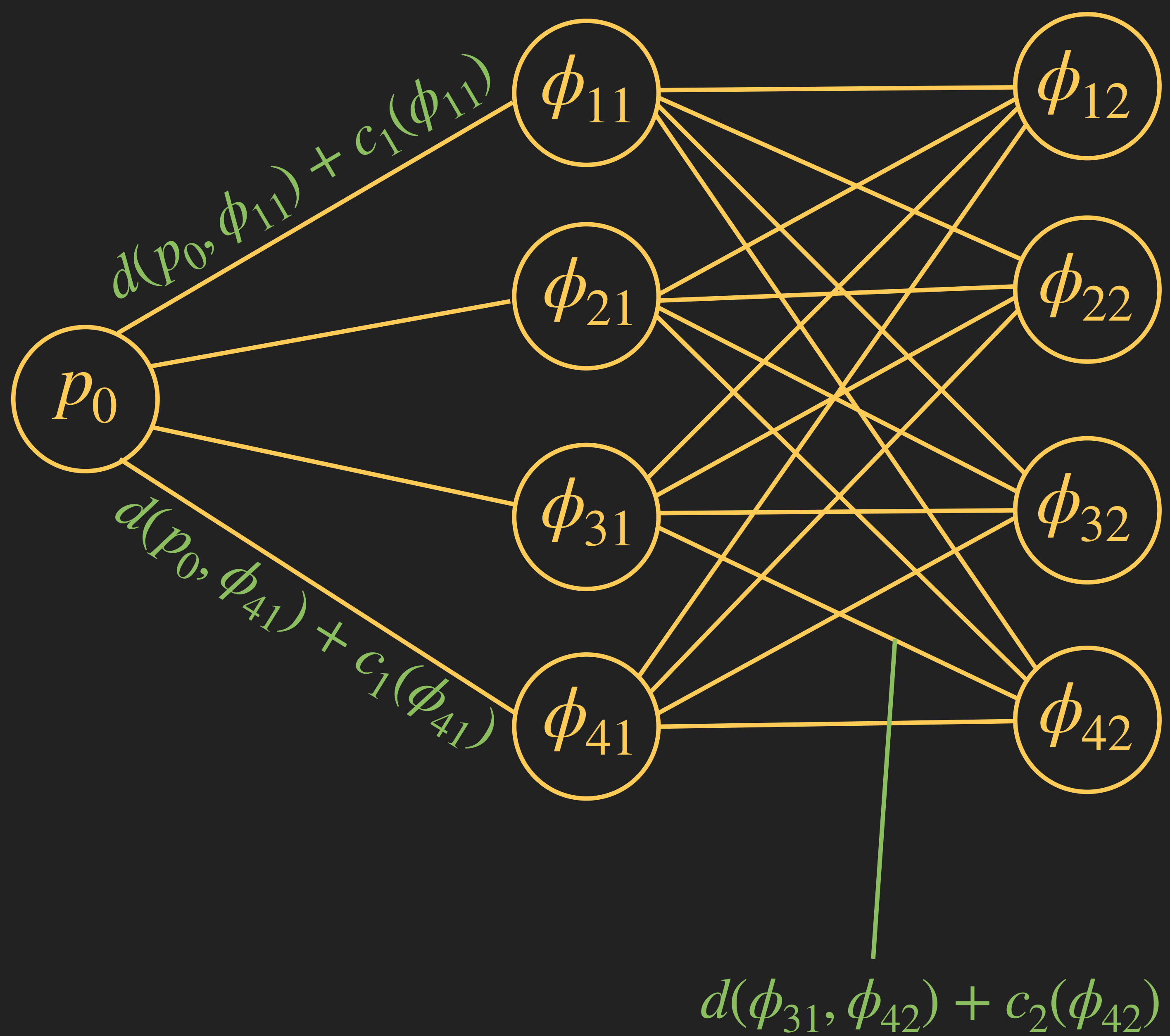
$$O\left(\frac{\epsilon^2}{\log k} \cdot \frac{\mathsf{DYN}}{\mathsf{diam}}\right)$$

if all suggestions queried

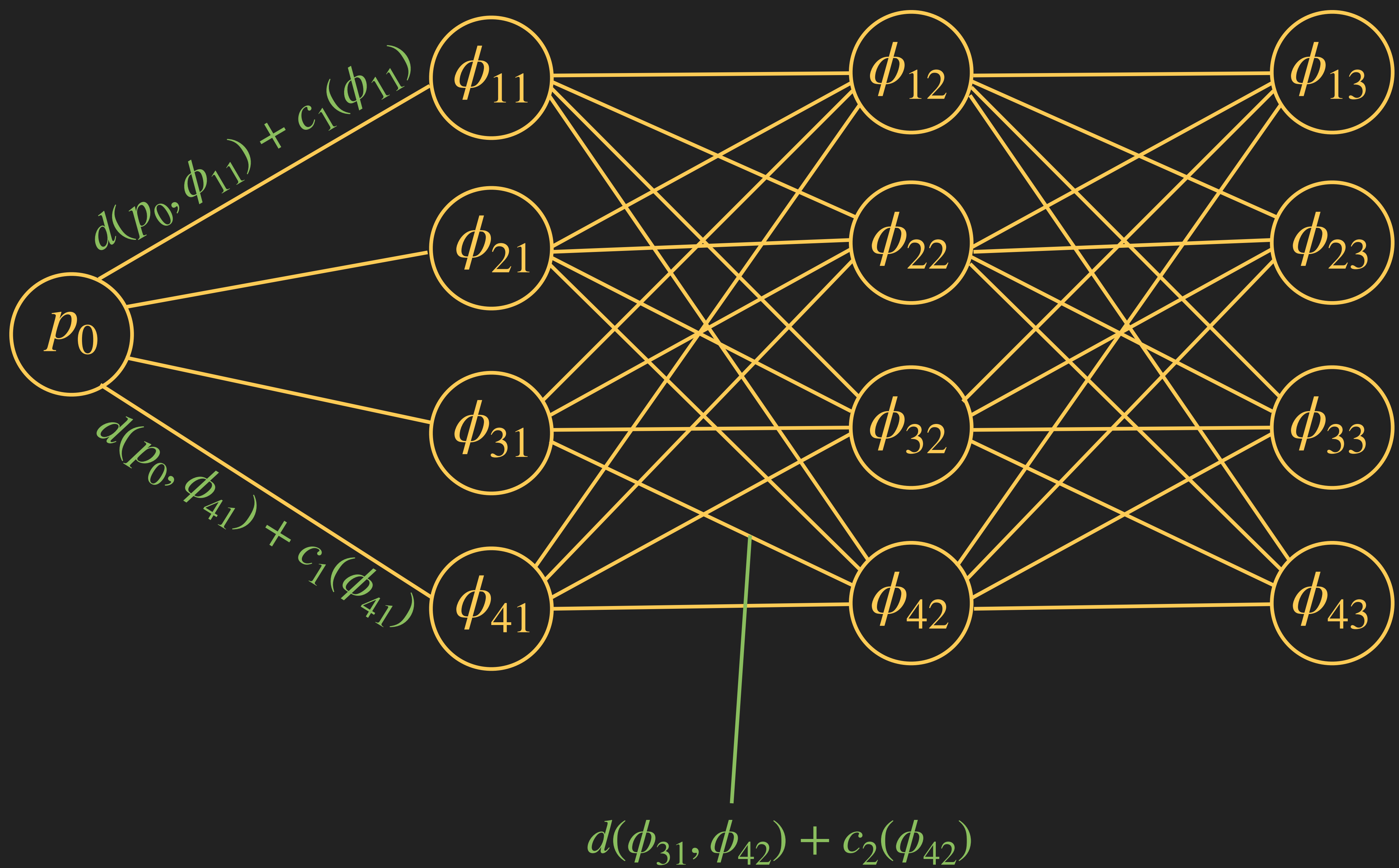
$$\tilde{O}\left(\frac{\epsilon^3}{k} \cdot \frac{\mathsf{DYN}}{\mathsf{diam}}\right)$$

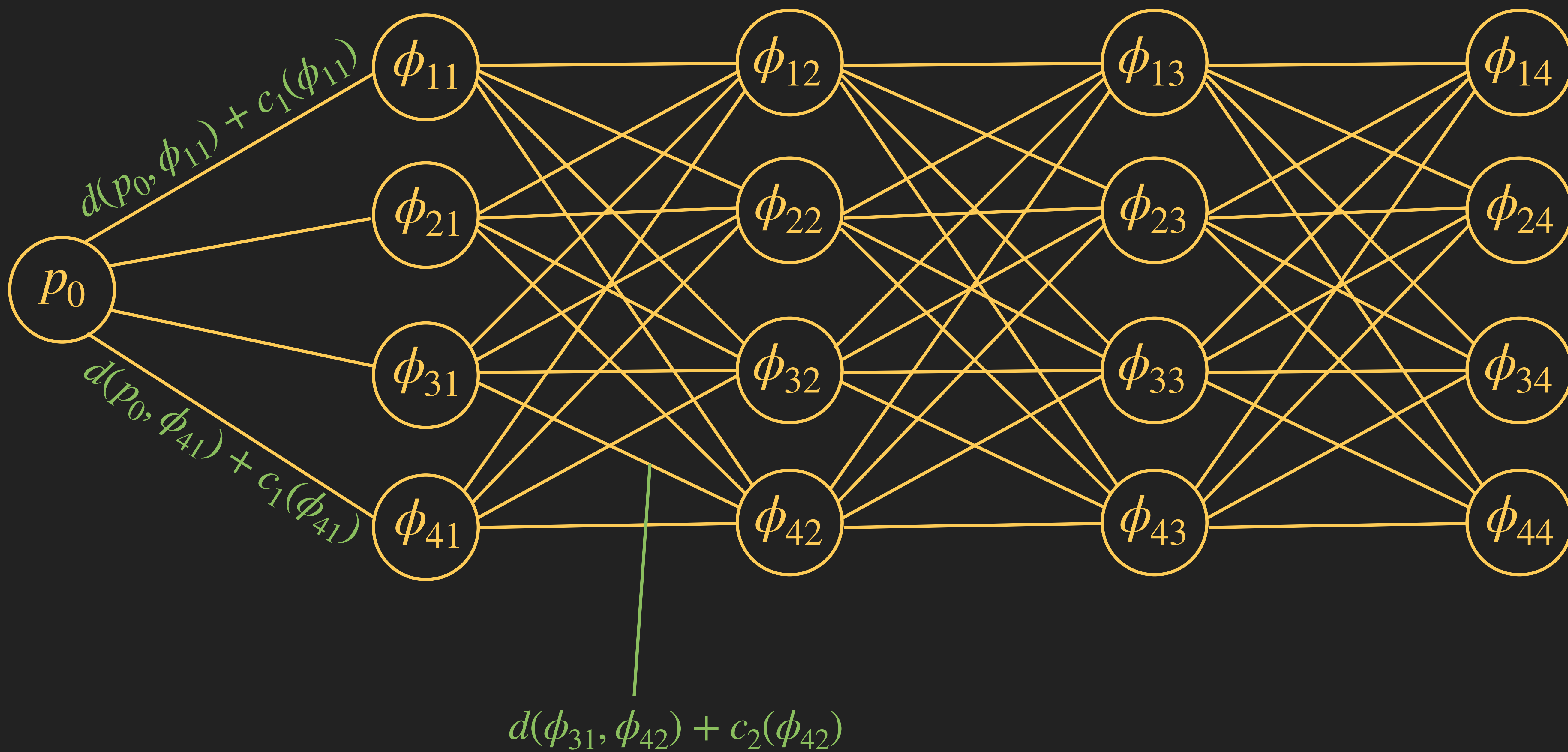
if one suggestion queried per time step


p_0
ϕ_{11}
ϕ_{21}
ϕ_{31}
ϕ_{41}
$d(p_0, \phi_{11}) + c_1(\phi_{11})$
$d(p_0, \phi_{41}) + c_1(\phi_{41})$



p_0
ϕ_{11}
ϕ_{21}
ϕ_{31}
ϕ_{41}
ϕ_{12}
ϕ_{22}
ϕ_{32}
ϕ_{42}
$d(p_0, \phi_{11}) + c_1(\phi_{11})$
$d(p_0, \phi_{41}) + c_1(\phi_{41})$
$d(\phi_{31}, \phi_{42}) + c_2(\phi_{42})$


p_0
ϕ_{11}
ϕ_{21}
ϕ_{31}
ϕ_{41}
ϕ_{12}
ϕ_{22}
ϕ_{32}
ϕ_{42}
ϕ_{13}
ϕ_{23}
ϕ_{33}
ϕ_{43}
$d(p_0, \phi_{11}) + c_1(\phi_{11})$
$d(p_0, \phi_{41}) + c_1(\phi_{41})$
$d(\phi_{31}, \phi_{42}) + c_2(\phi_{42})$



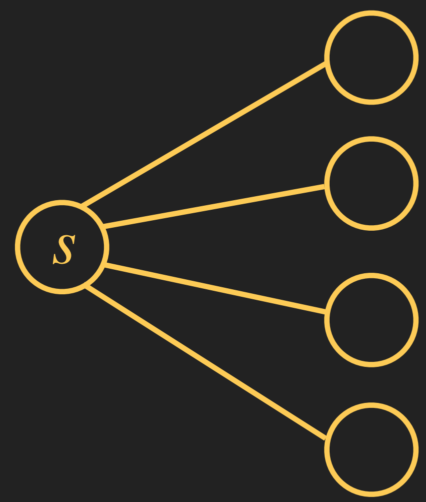
p_0
ϕ_{11}
ϕ_{21}
ϕ_{31}
ϕ_{41}
ϕ_{12}
ϕ_{22}
ϕ_{32}
ϕ_{42}
ϕ_{13}
ϕ_{23}
ϕ_{33}
ϕ_{43}
ϕ_{14}
ϕ_{24}
ϕ_{34}
ϕ_{44}
$d(p_0, \phi_{11}) + c_1(\phi_{11})$
$d(p_0, \phi_{41}) + c_1(\phi_{41})$
$d(\phi_{31}, \phi_{42}) + c_2(\phi_{42})$

Shortest Paths without a Map, but with an Entropic Regularizer

JOINT WORK WITH

SÉBASTIEN BUBECK (MICROSOFT RESEARCH)
YUVAL RABANI (HEBREW UNIVERSITY OF JERUSALEM)

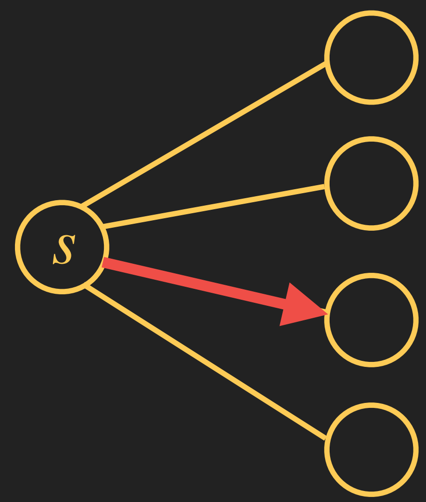
Layered Graph Traversal (LGT)



[Papadimitriou,Yannakakis 1989:
“Shortest Paths without a Map”]

- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed

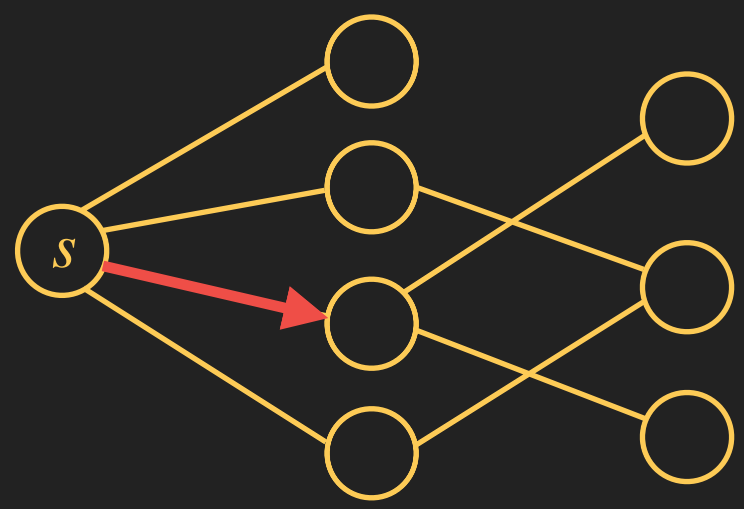
Layered Graph Traversal (LGT)



[Papadimitriou,Yannakakis 1989: "Shortest Paths without a Map"]

- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed

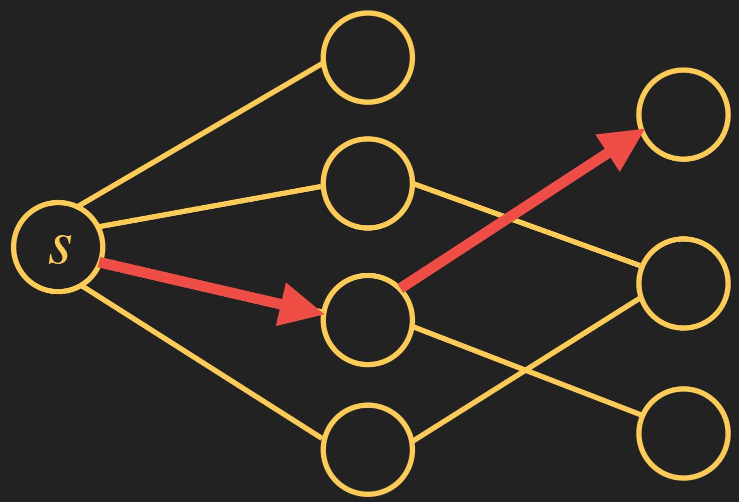
Layered Graph Traversal (LGT)



[Papadimitriou, Yannakakis 1989: "Shortest Paths without a Map"]

- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed

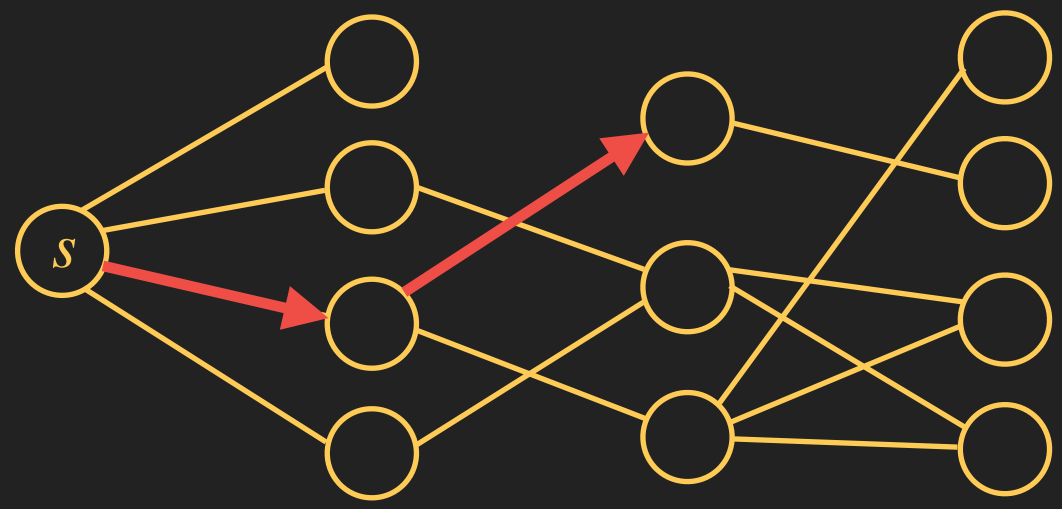
Layered Graph Traversal (LGT)



[Papadimitriou, Yannakakis 1989: "Shortest Paths without a Map"]

- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed

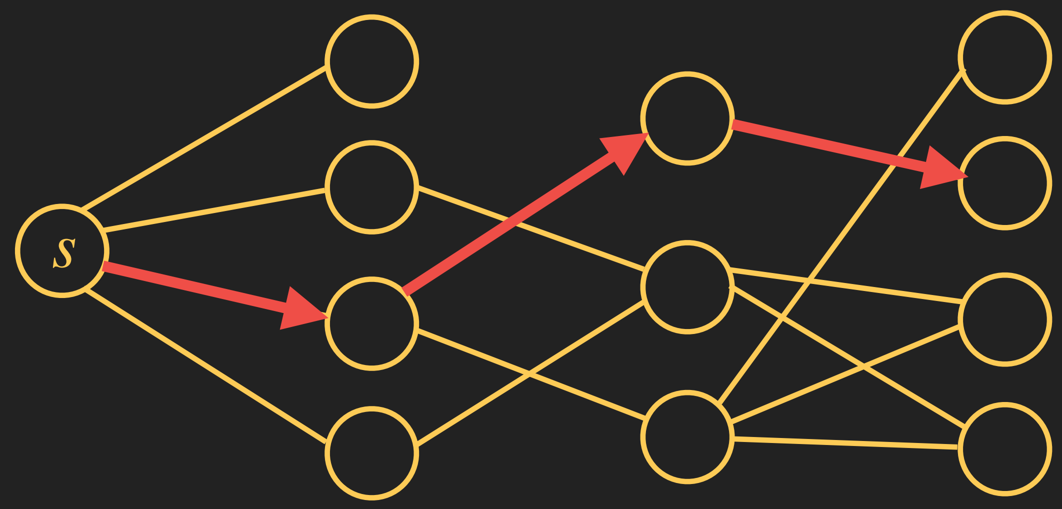
Layered Graph Traversal (LGT)



[Papadimitriou,Yannakakis 1989:
"Shortest Paths without a Map"]

- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed

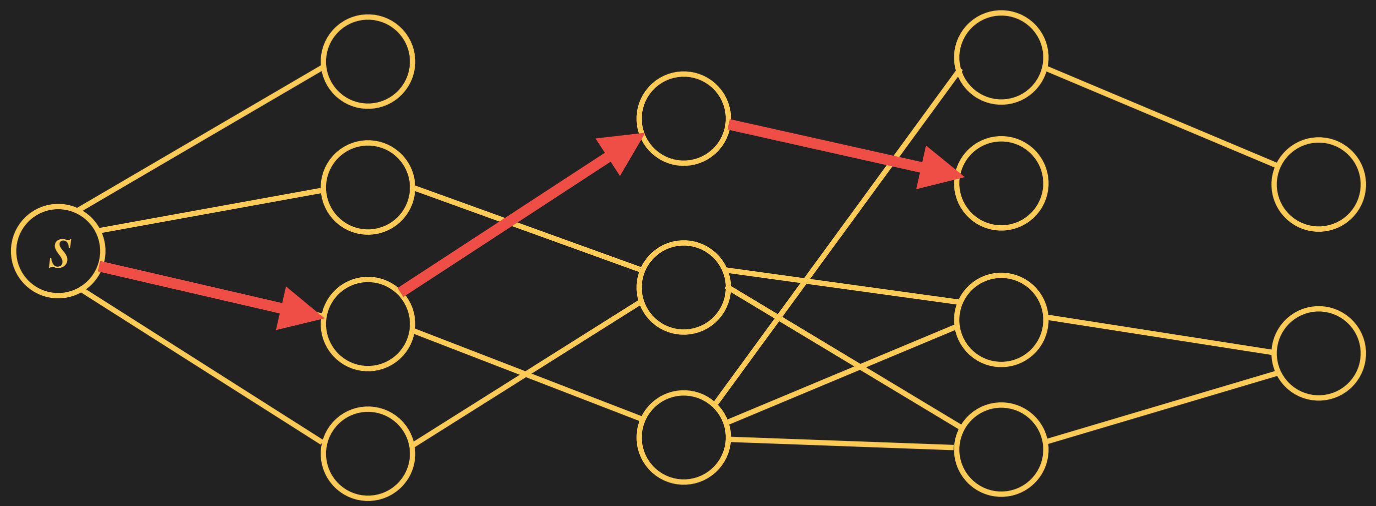
Layered Graph Traversal (LGT)



[Papadimitriou,Yannakakis 1989:
"Shortest Paths without a Map"]

- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed

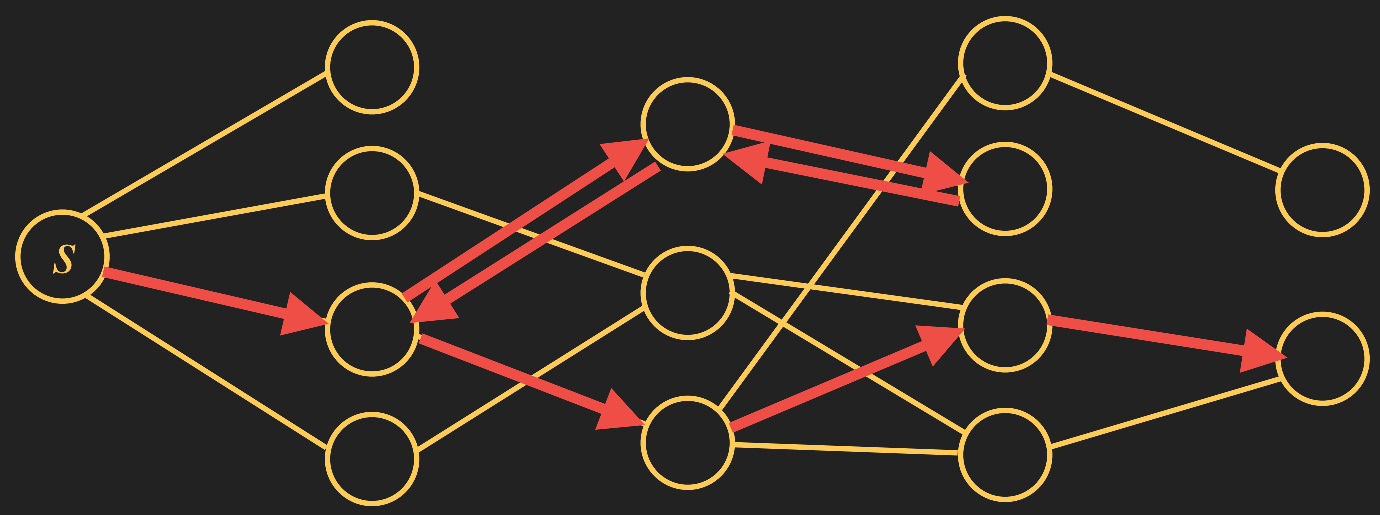
Layered Graph Traversal (LGT)



[Papadimitriou,Yannakakis 1989: "Shortest Paths without a Map"]

- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed

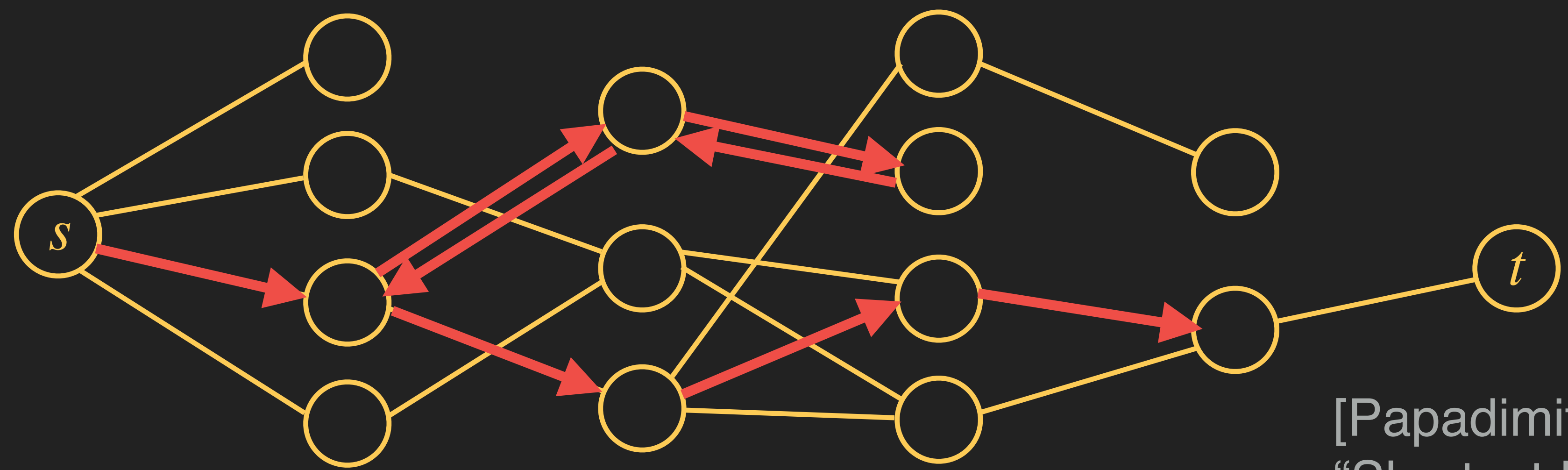
Layered Graph Traversal (LGT)



[Papadimitriou,Yannakakis 1989: “Shortest Paths without a Map”]

- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed

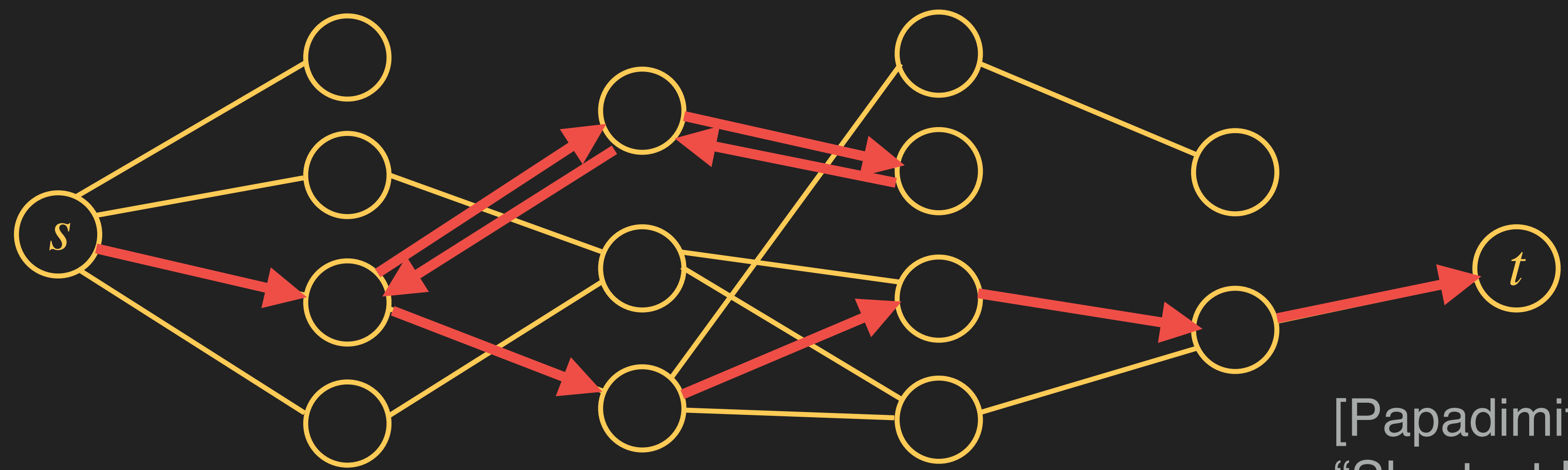
Layered Graph Traversal (LGT)



[Papadimitriou, Yannakakis 1989: "Shortest Paths without a Map"]

- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed

Layered Graph Traversal (LGT)



[Papadimitriou,Yannakakis 1989: "Shortest Paths without a Map"]

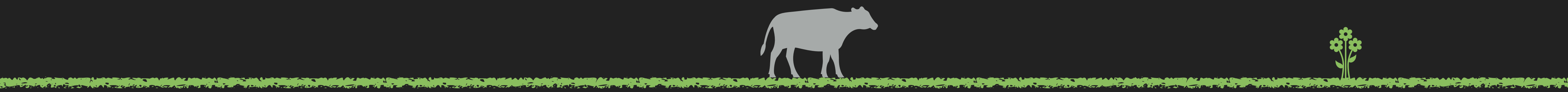
- Vertices in layers $L_0 = \{s\}, L_1, L_2, \ldots, L_T = \{t\}$
- Weighted edges between adjacent layers
- Searcher starts at s
- When L_i reached: L_{i+1} and edges between L_i, L_{i+1} revealed
- cost = distance traveled until reaching t
- only parameter: $k := \max_i |L_i|$

Chasing Small Sets (aka Metrical Service Systems)

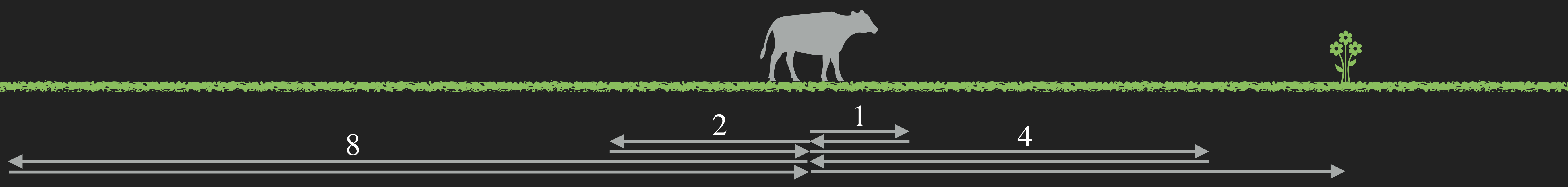
- 1 server in metric space M
- At time $t = 1,2,\ldots$
  - Set $S_t \subset M$ requested, $|S_t| \leq k$
  - Server must move to S_t
- Cost = distance moved

Theorem [Fiat et al. '91]: This problem is equivalent to LGT

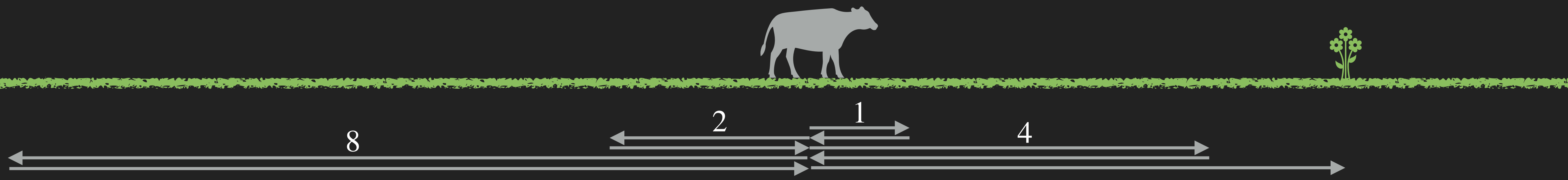
Simplest Version: Cow Path Problem



Simplest Version: Cow Path Problem

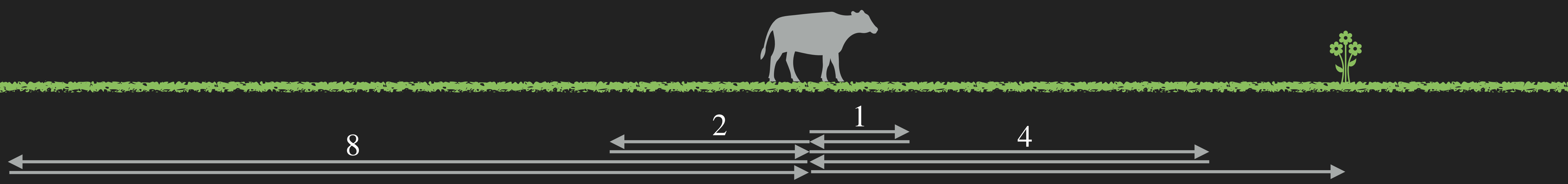


Simplest Version: Cow Path Problem

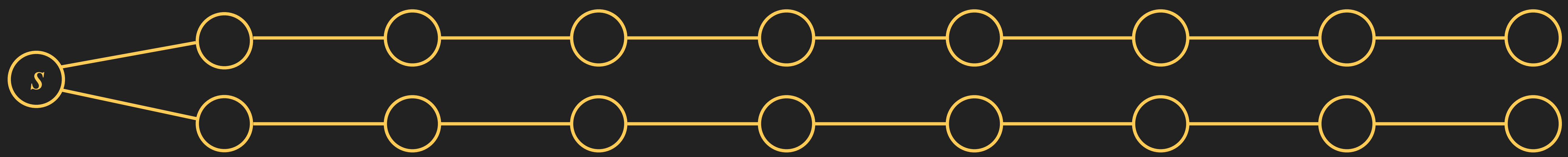


Doubling strategy 9-competitive (best deterministic algo)

Simplest Version: Cow Path Problem



Doubling strategy 9-competitive (best deterministic algo)



State of the Art:

Old Bounds:

- deterministic: $O(k \cdot 2^k) \cap \Omega(2^k)$-competitive [Burley '96, Fiat et al. '91]
- randomized: $O(k^{13}) \cap \Omega(k^2/\log^{1+\epsilon} k)$ [Ramesh '93]
  - stuck since 1993

State of the Art:

Old Bounds:

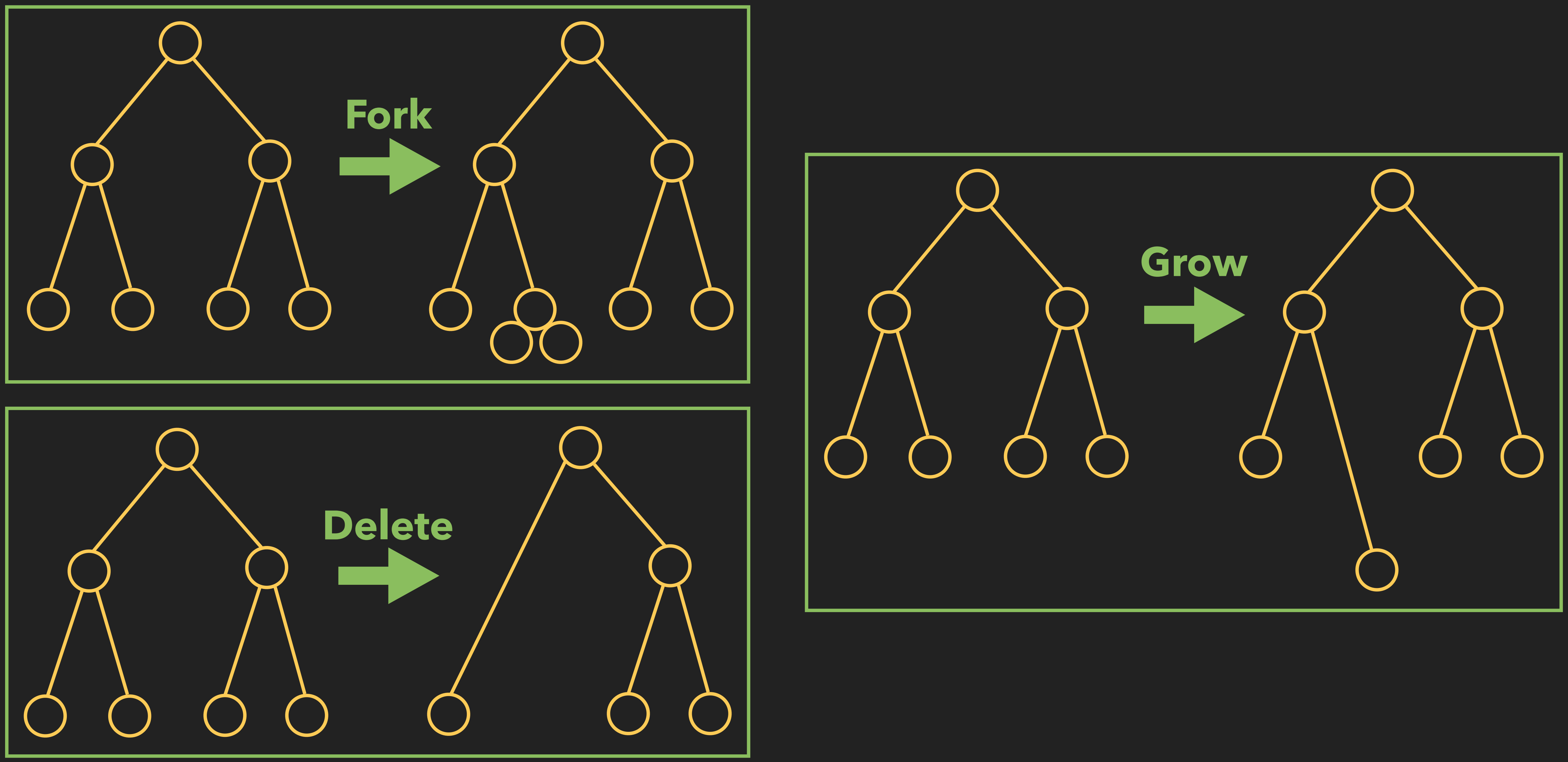
- deterministic: $O(k \cdot 2^k) \cap \Omega(2^k)$-competitive [Burley '96, Fiat et al. '91]
- randomized: $O(k^{13}) \cap \Omega(k^2/\log^{1+\epsilon} k)$ [Ramesh '93]
  - stuck since 1993

New tight randomized bound: $\Theta(k^2)$

[Bubeck-Coester-Rabani 22,23]

Evolving Tree Game (ETG)

- Binary tree evolves over time:



- Agent must stay at leaves

- Cost = distance moved by agent

Reduction: LGT $\leq$ ETG

- Wlog layered graph is a tree [Fiat et al. '91]

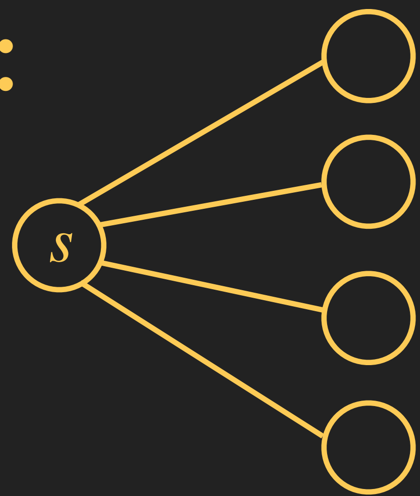
  (build online the tree of shortest paths from s)

Reduction: LGT $\leq$ ETG

- Wlog layered graph is a tree [Fiat et al. '91]

(build online the tree of shortest paths from s)

LGT:



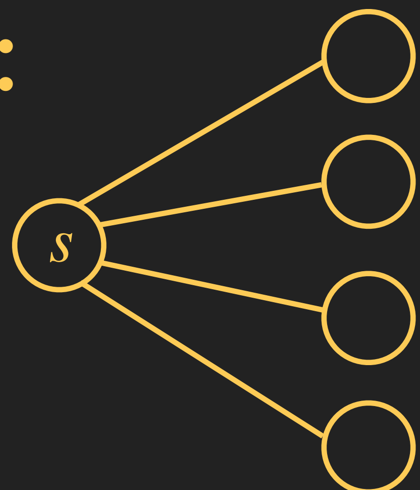
ETG:

Reduction: LGT $\leq$ ETG

- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



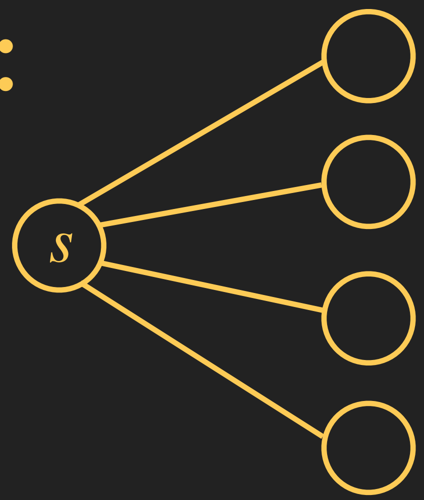
ETG:

Reduction: LGT $\leq$ ETG

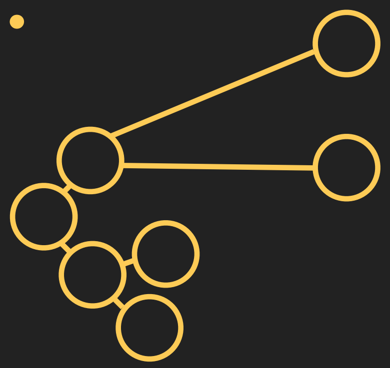
- Wlog layered graph is a tree [Fiat et al. '91]

(build online the tree of shortest paths from s)

LGT:



ETG:

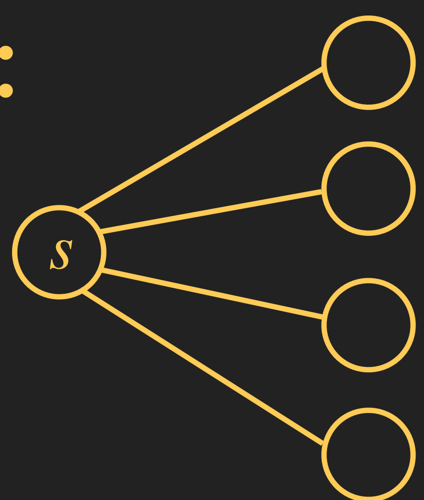


Reduction: LGT $\leq$ ETG

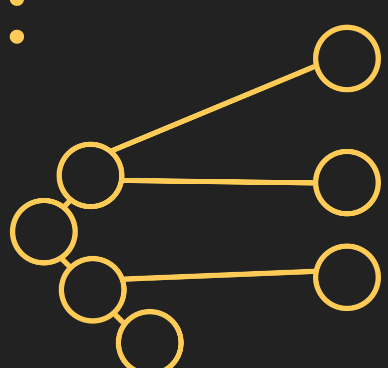
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

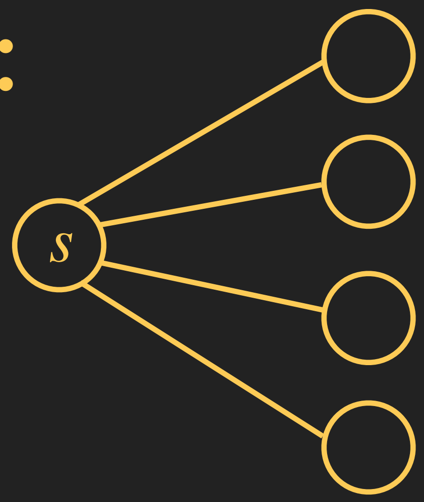


Reduction: LGT $\leq$ ETG

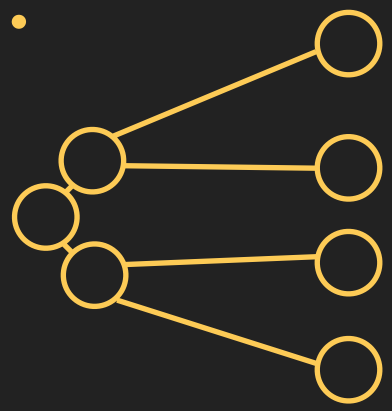
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

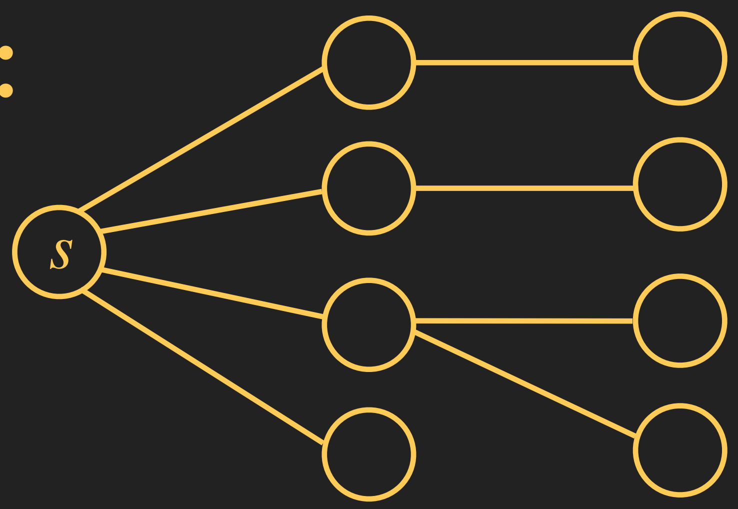


Reduction: LGT $\leq$ ETG

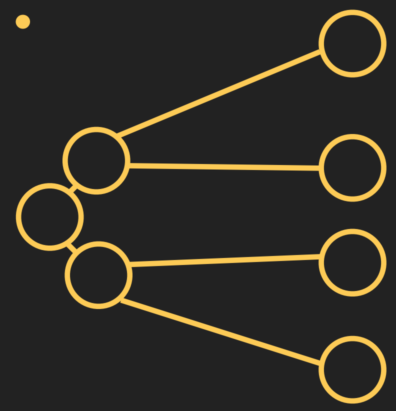
- Wlog layered graph is a tree [Fiat et al. '91]

(build online the tree of shortest paths from s)

LGT:



ETG:

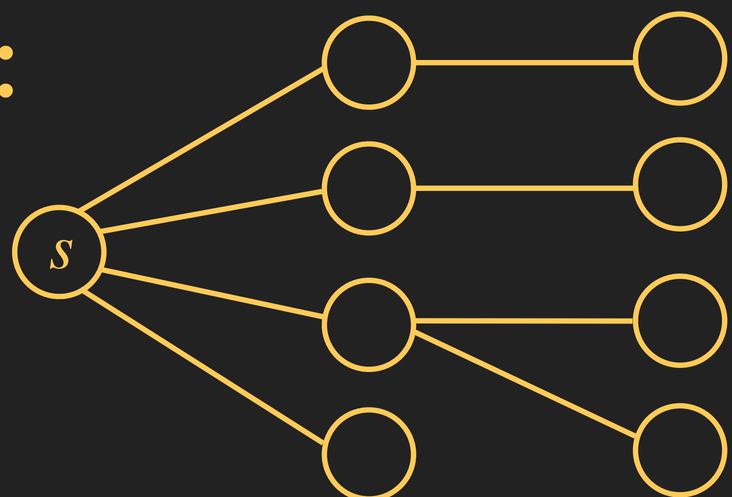


Reduction: LGT $\leq$ ETG

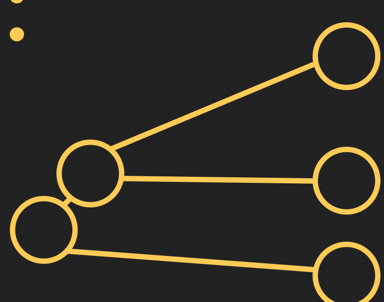
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

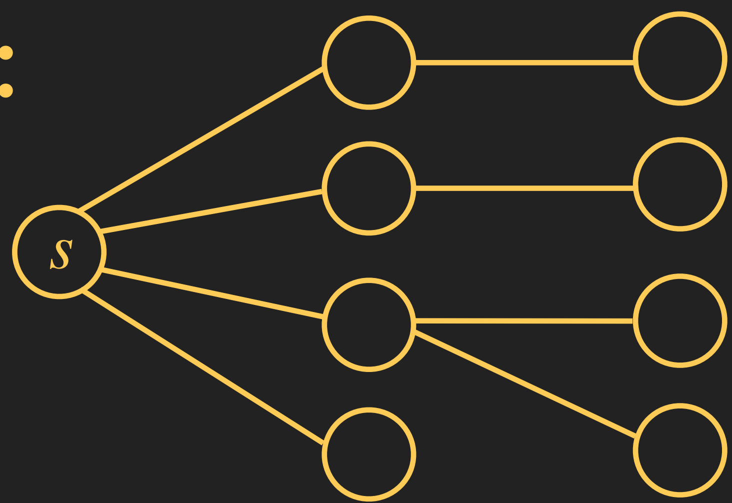


Reduction: LGT $\leq$ ETG

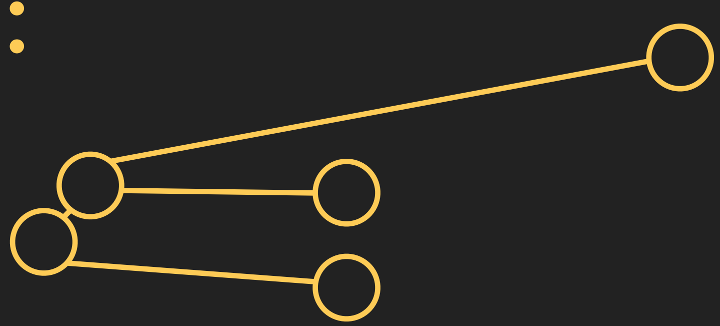
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

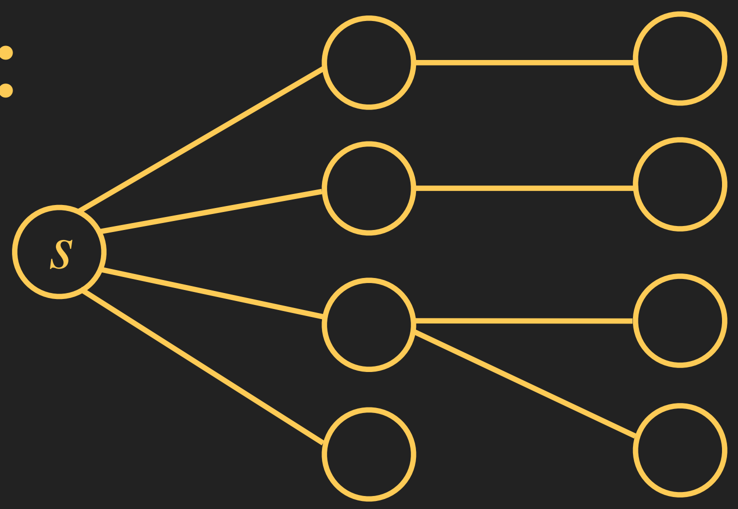


Reduction: LGT $\leq$ ETG

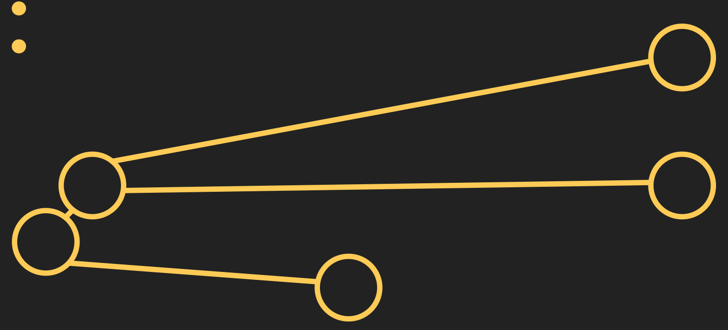
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

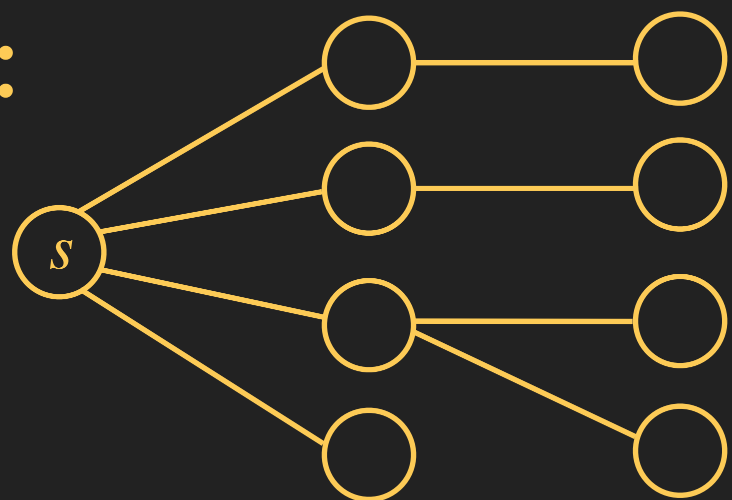


Reduction: LGT $\leq$ ETG

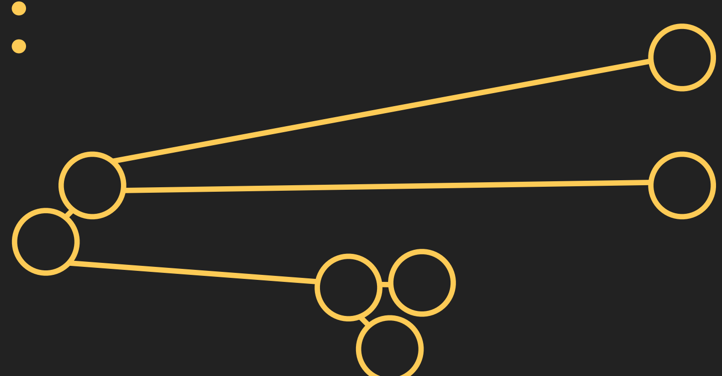
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

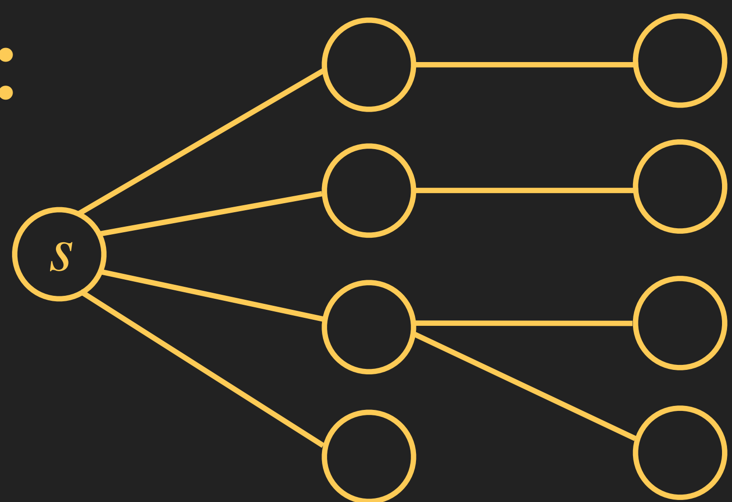


Reduction: LGT $\leq$ ETG

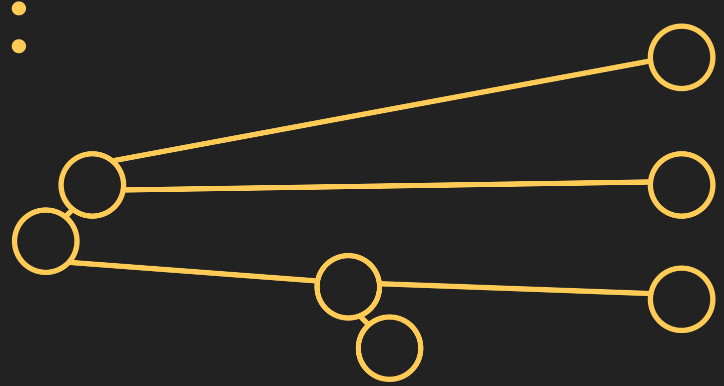
- Wlog layered graph is a tree [Fiat et al. '91]

(build online the tree of shortest paths from s)

LGT:



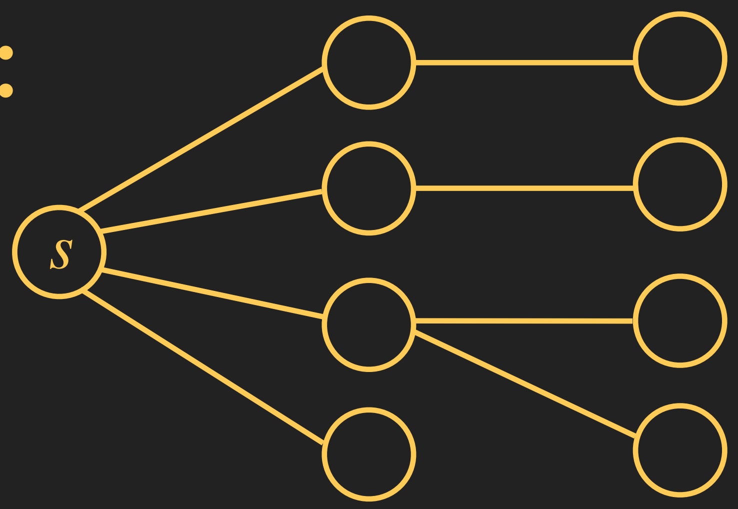
ETG:



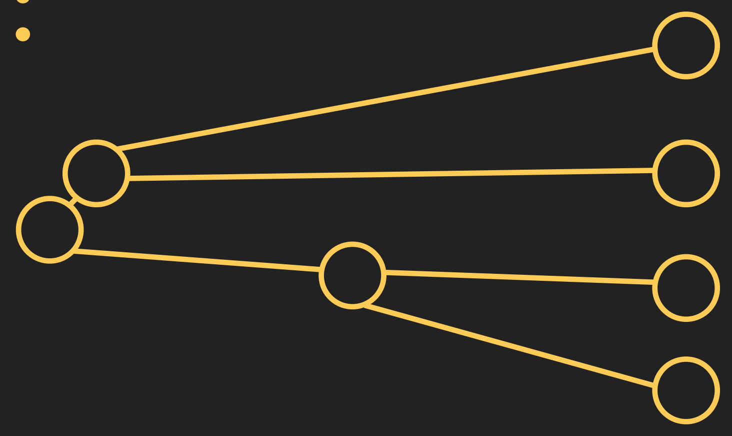
Reduction: LGT $\leq$ ETG

- Wlog layered graph is a tree [Fiat et al. '91]
  (build online the tree of shortest paths from s)

LGT:



ETG:

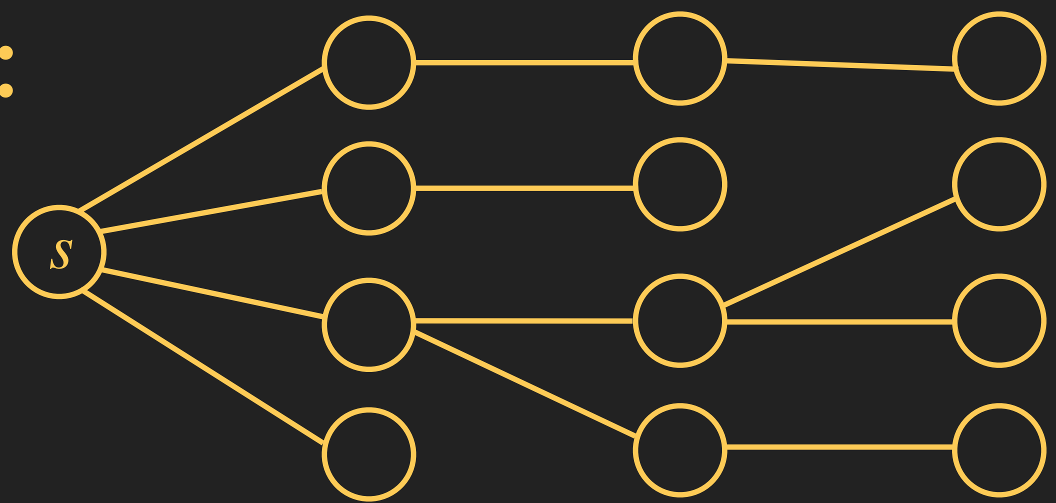


Reduction: LGT $\leq$ ETG

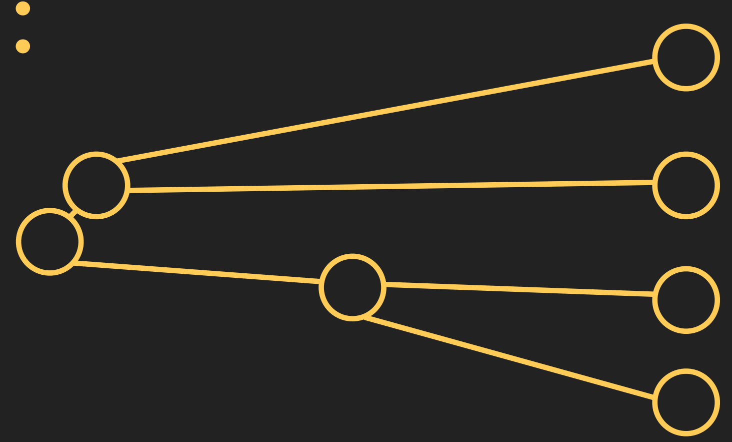
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

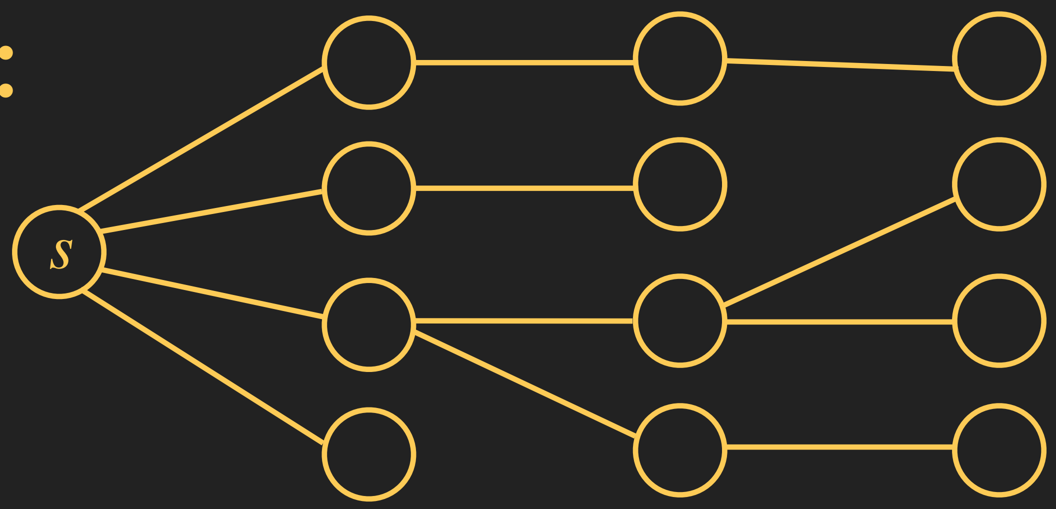


Reduction: LGT $\leq$ ETG

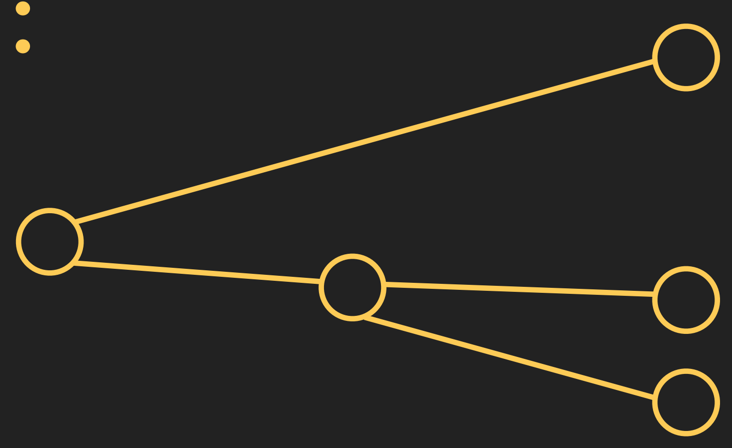
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

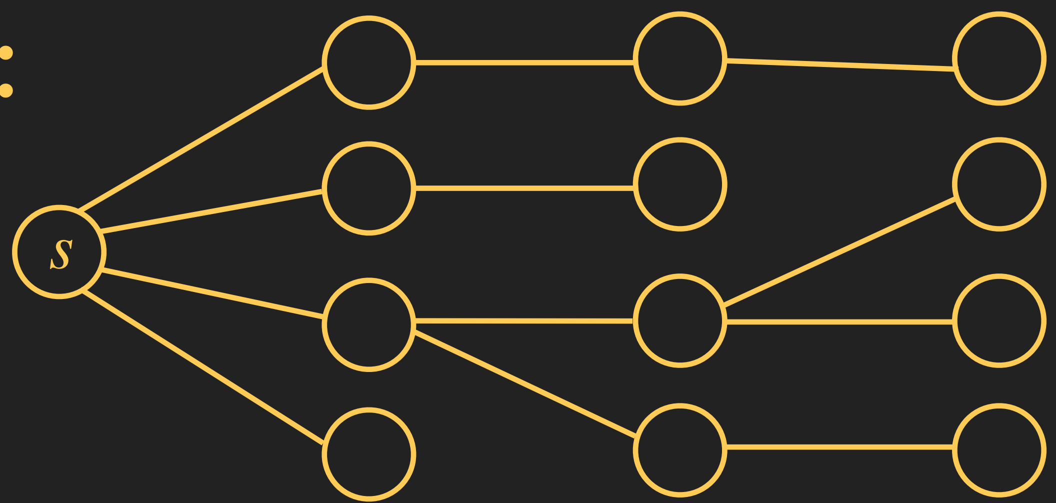


Reduction: LGT $\leq$ ETG

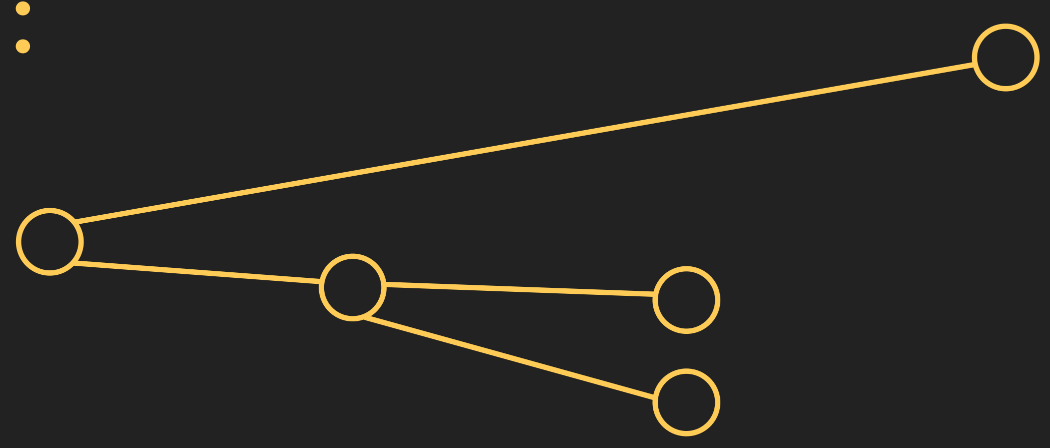
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



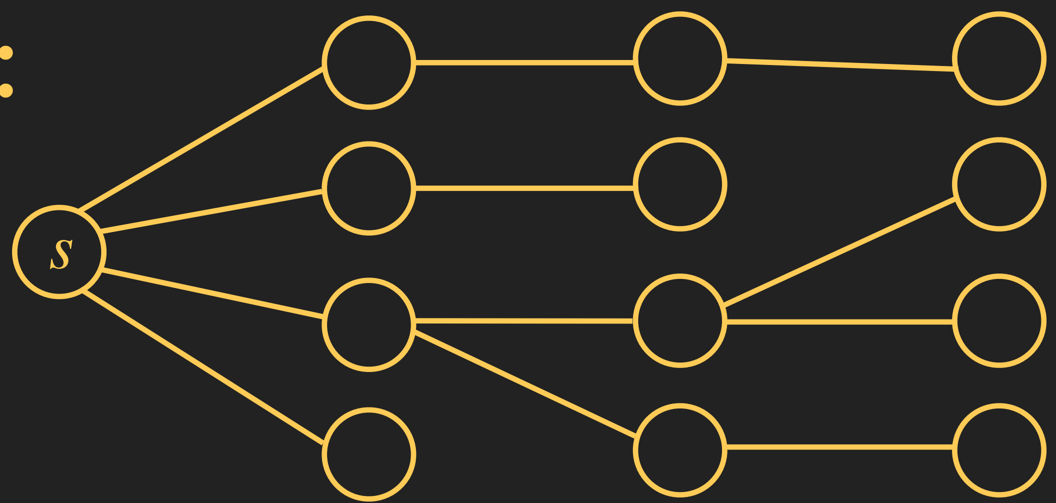
ETG:



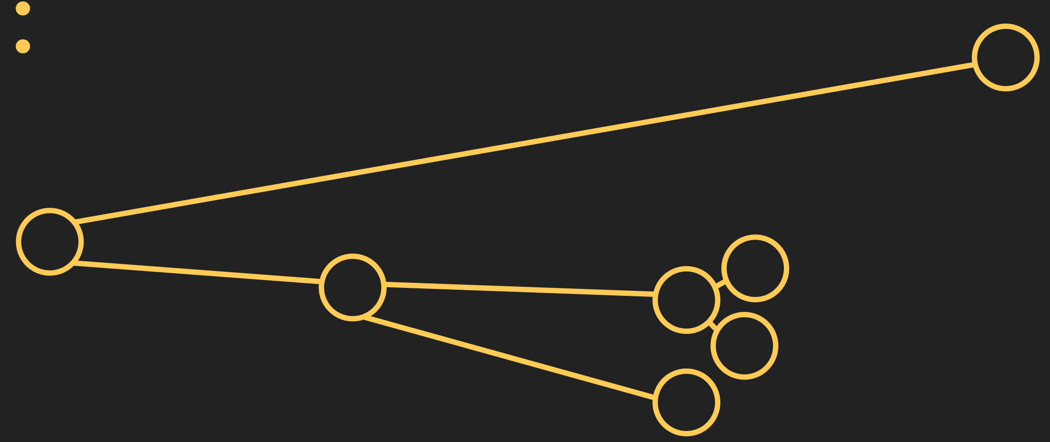
Reduction: LGT $\leq$ ETG

- Wlog layered graph is a tree [Fiat et al. '91]
  (build online the tree of shortest paths from s)

LGT:



ETG:

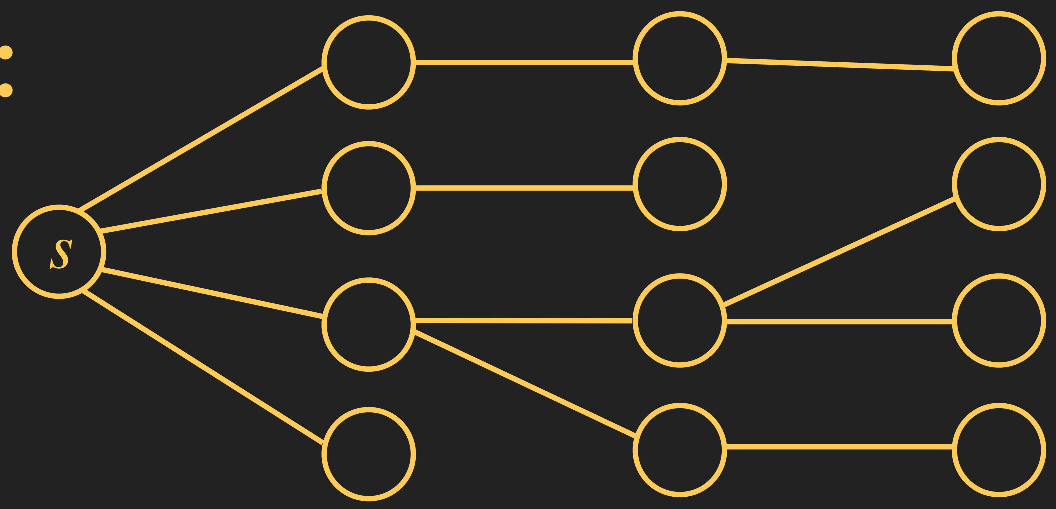


Reduction: LGT $\leq$ ETG

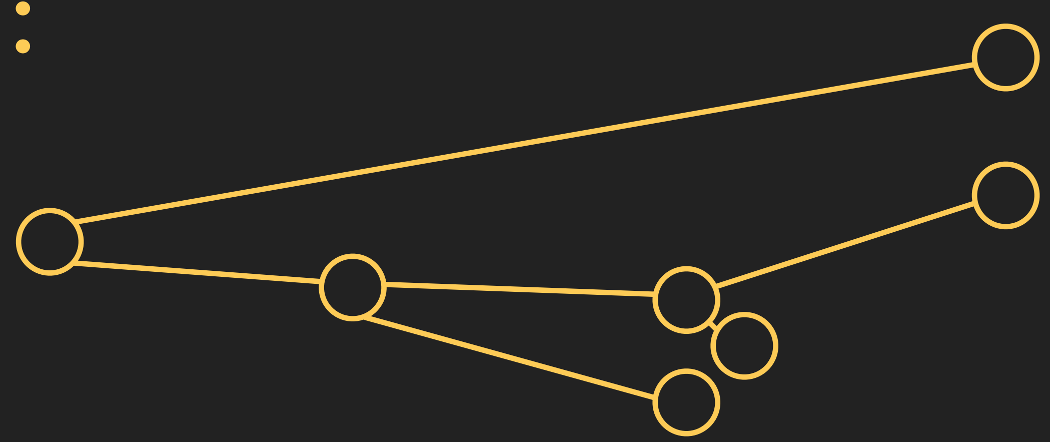
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

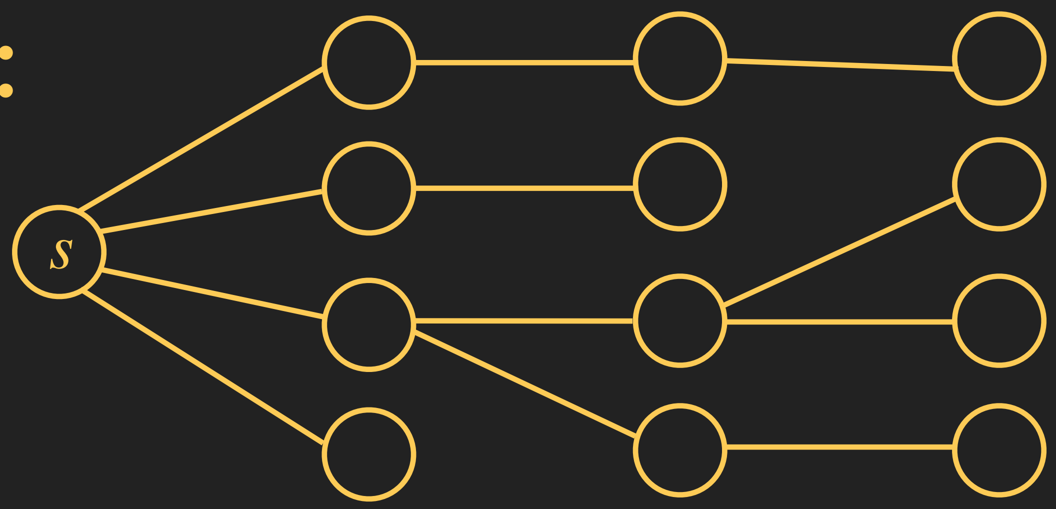


Reduction: LGT $\leq$ ETG

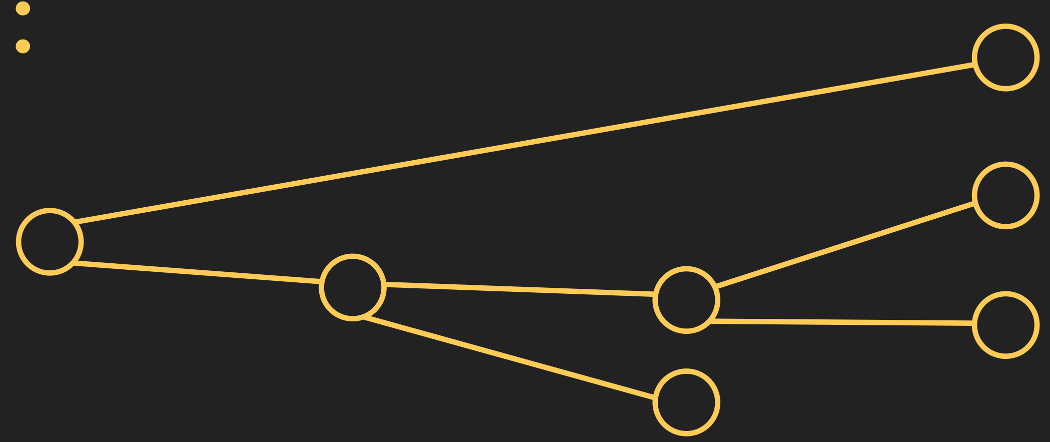
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

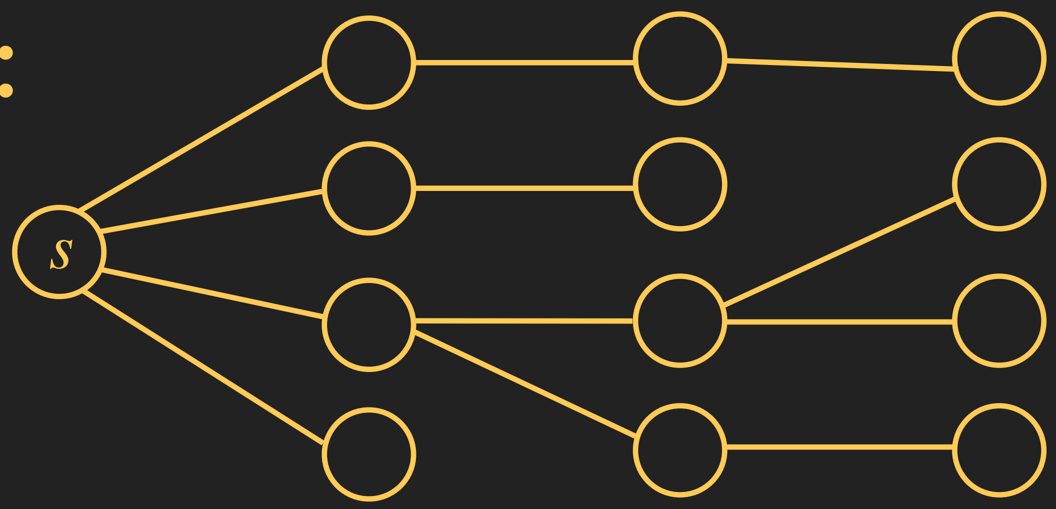


Reduction: LGT $\leq$ ETG

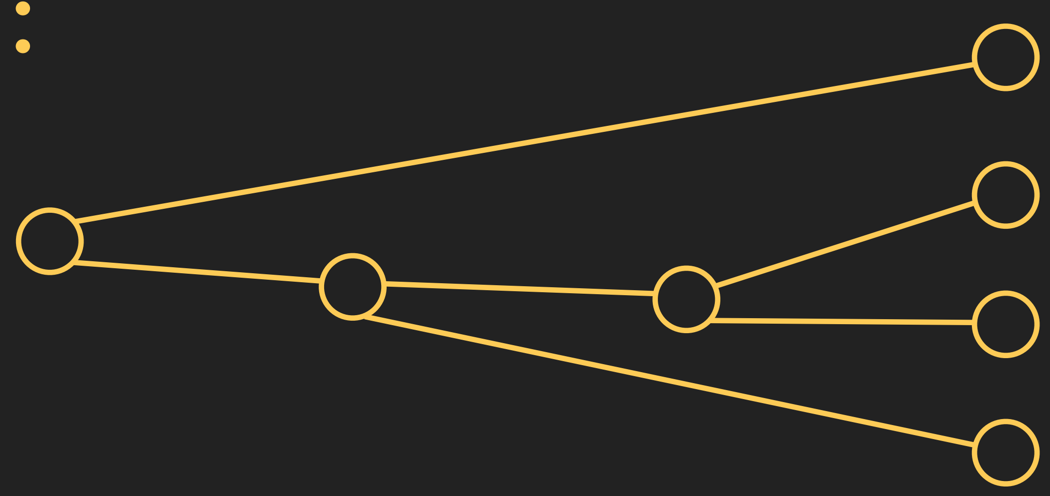
- Wlog layered graph is a tree [Fiat et al. '91]

(build online the tree of shortest paths from s)

LGT:



ETG:

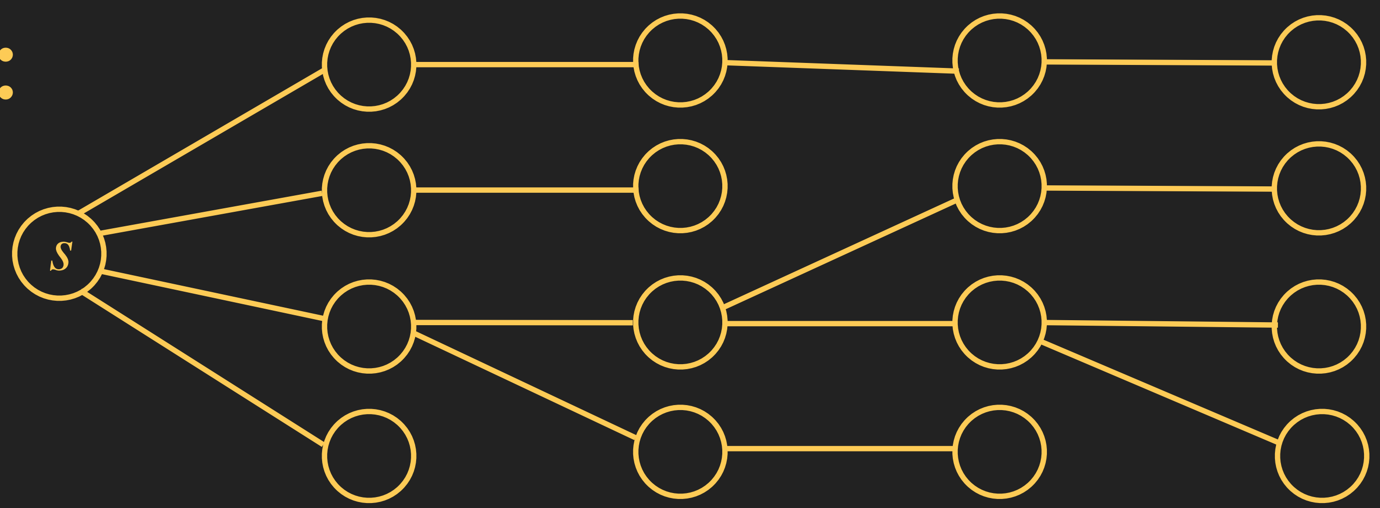


Reduction: LGT $\leq$ ETG

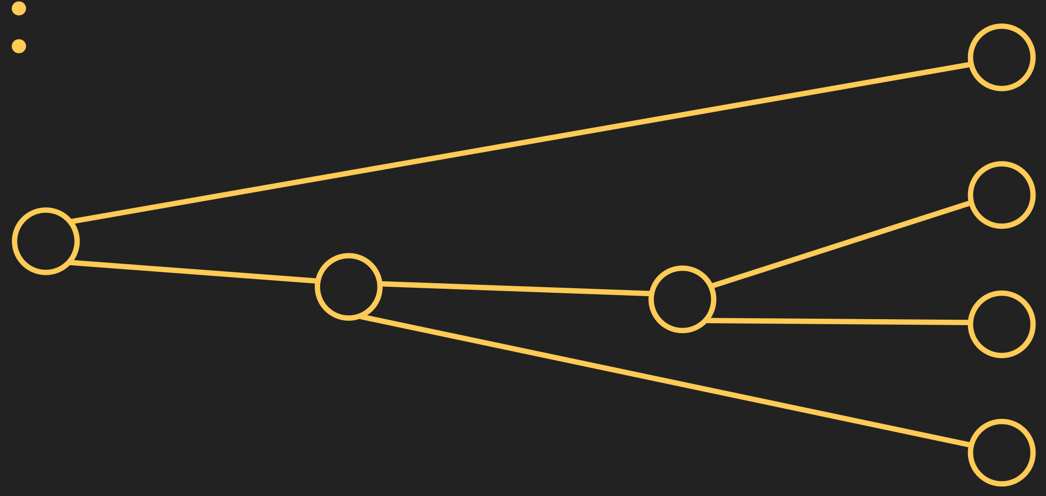
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

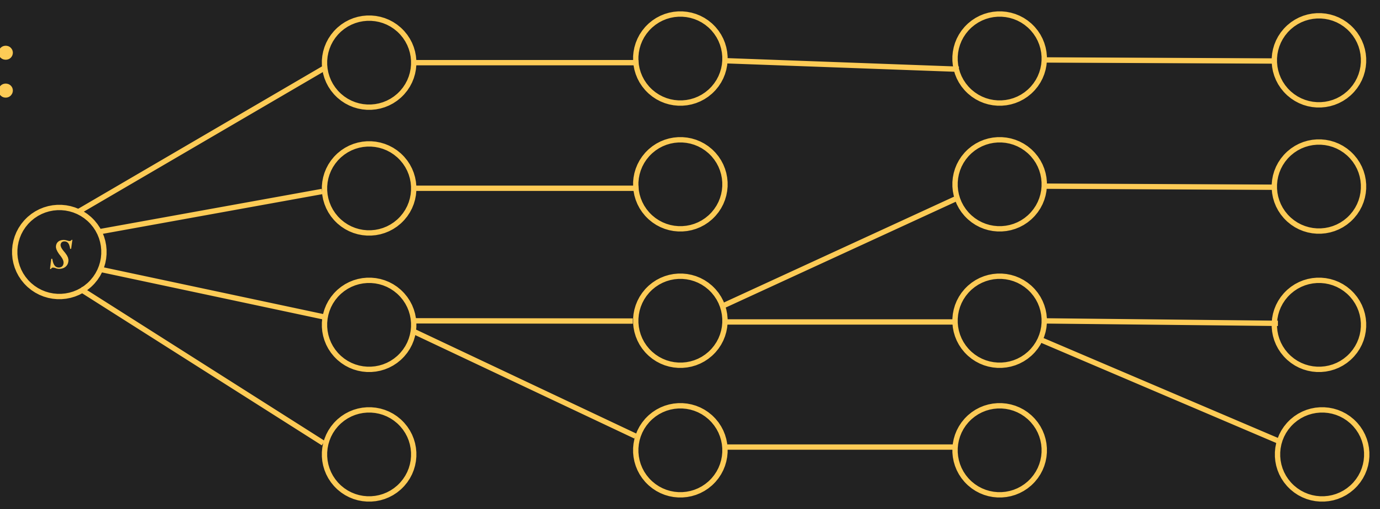


Reduction: LGT $\leq$ ETG

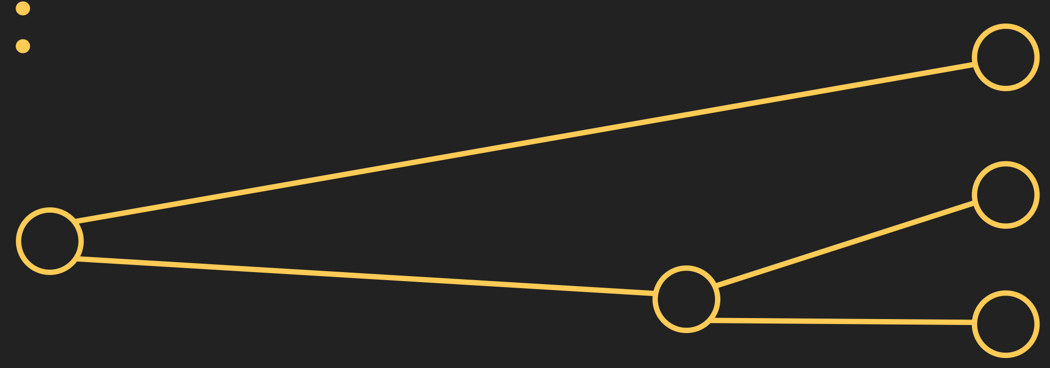
- Wlog layered graph is a tree [Fiat et al. '91]

(build online the tree of shortest paths from s)

LGT:



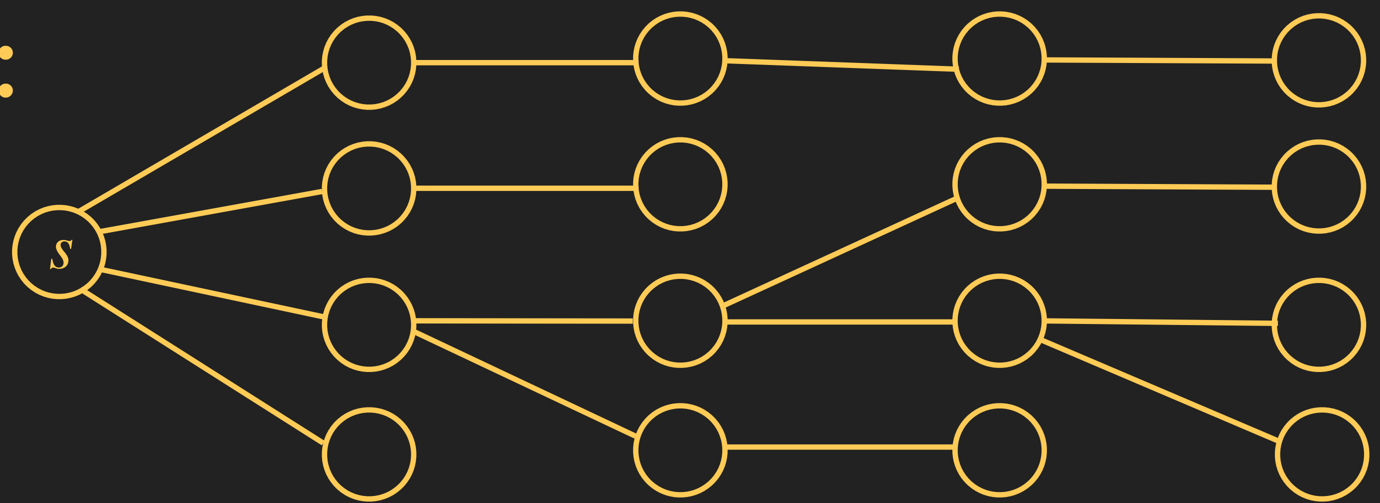
ETG:



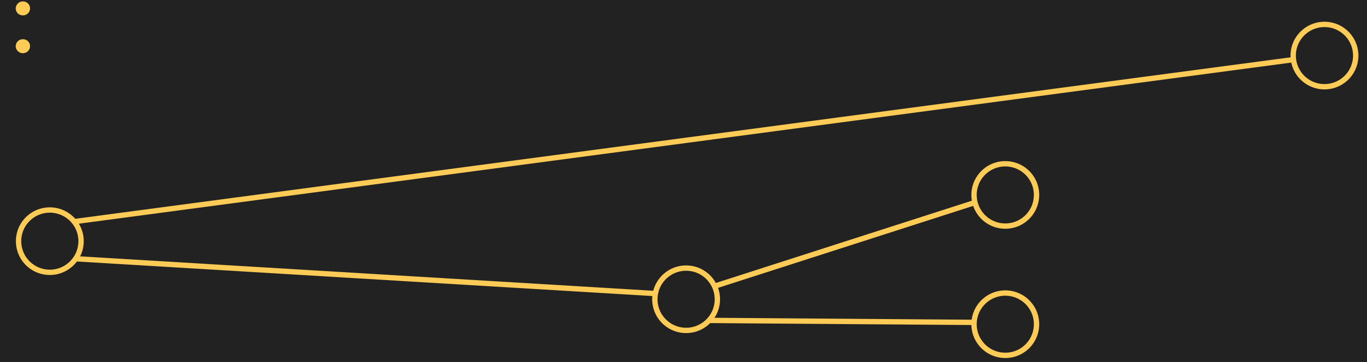
Reduction: LGT $\leq$ ETG

- Wlog layered graph is a tree [Fiat et al. '91]
  (build online the tree of shortest paths from s)

LGT:

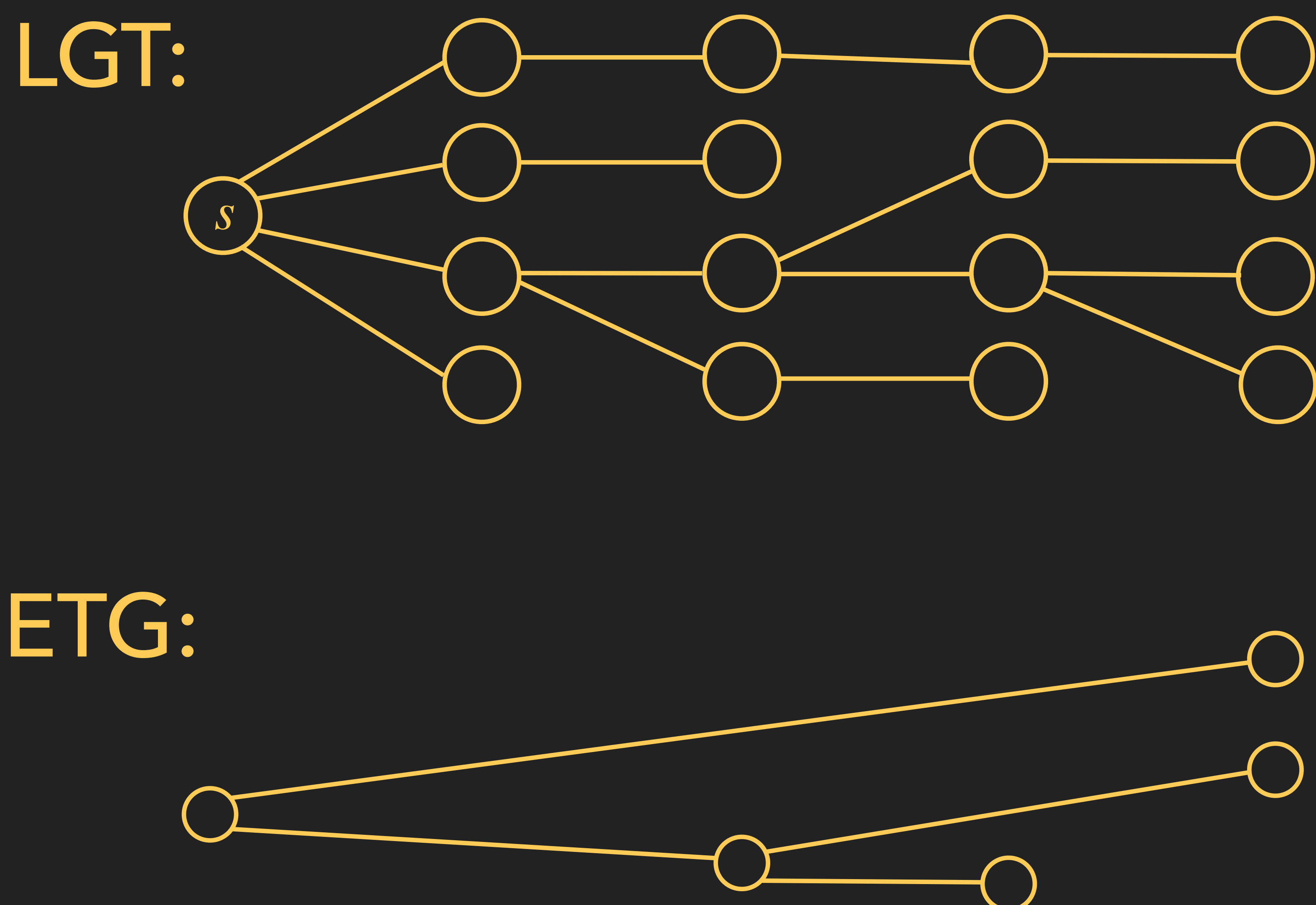


ETG:



Reduction: LGT $\leq$ ETG

- Wlog layered graph is a tree [Fiat et al. '91]
  (build online the tree of shortest paths from s)

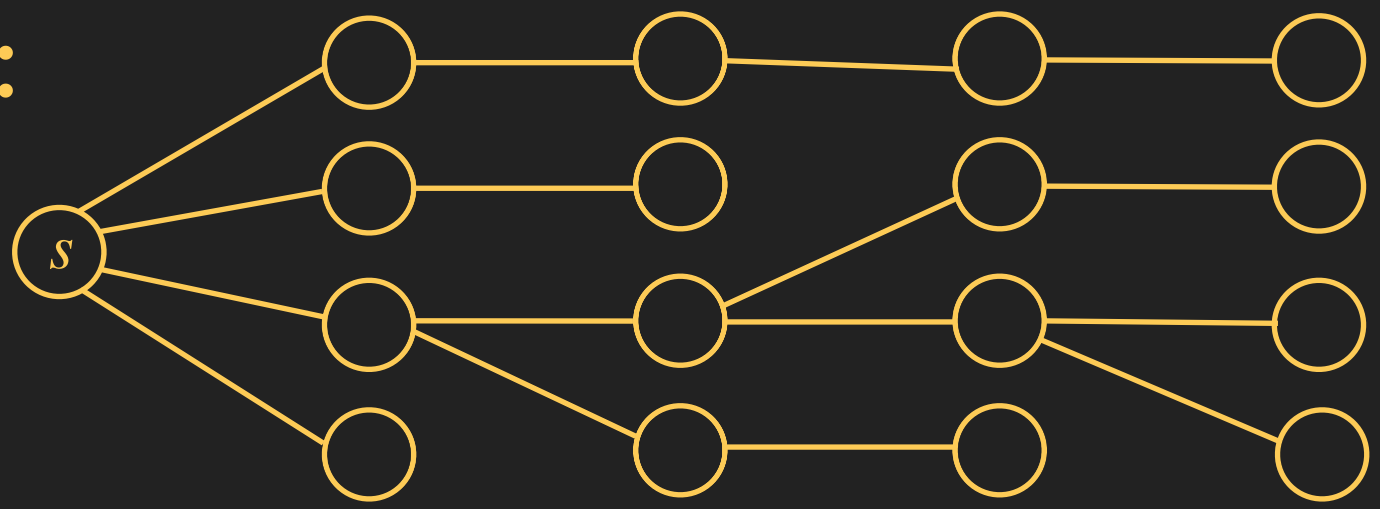


Reduction: LGT $\leq$ ETG

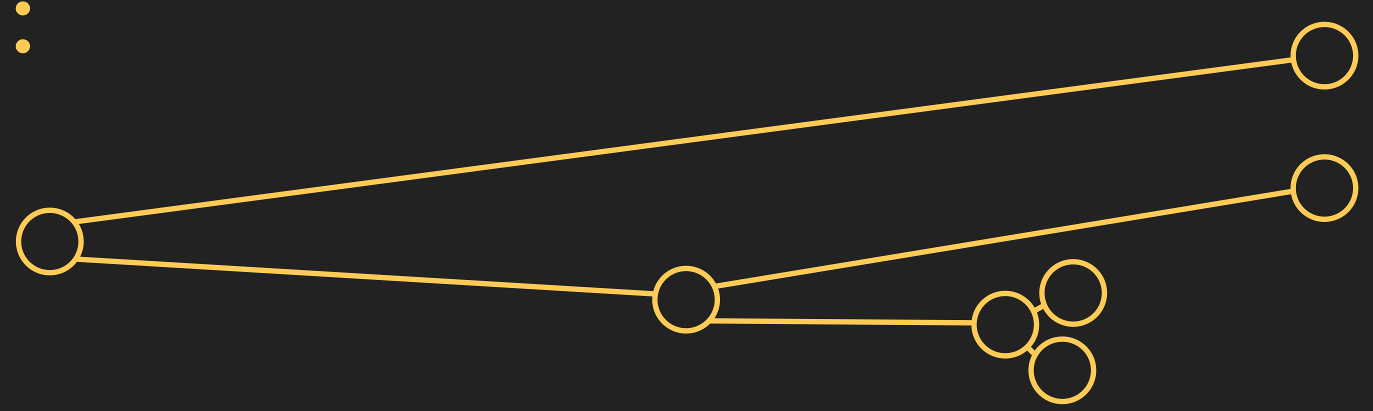
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

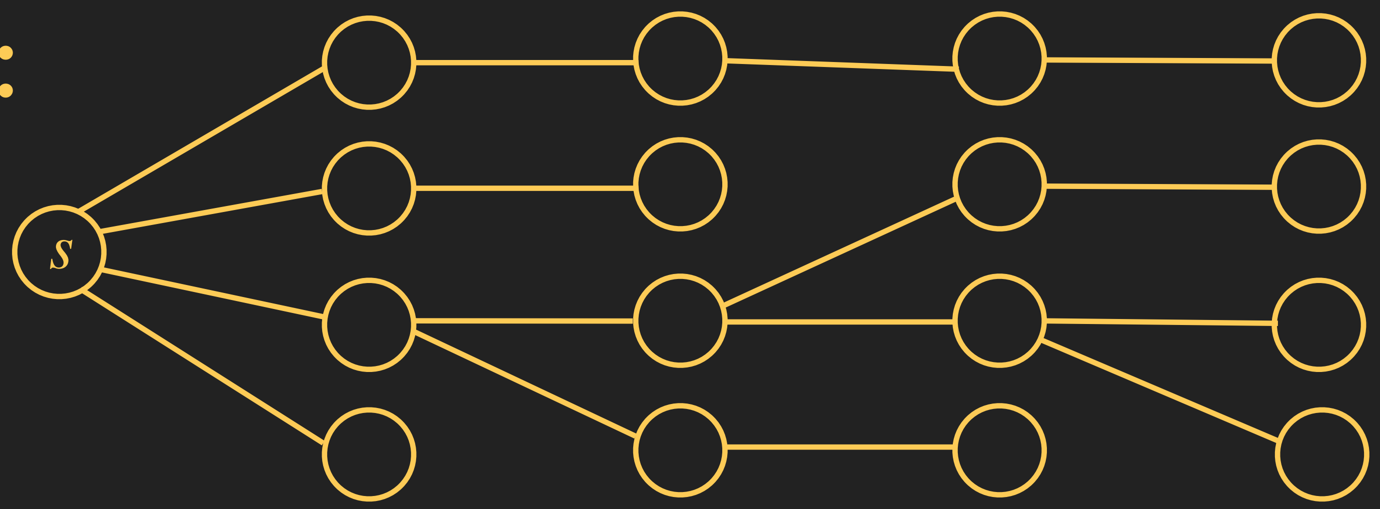


Reduction: LGT $\leq$ ETG

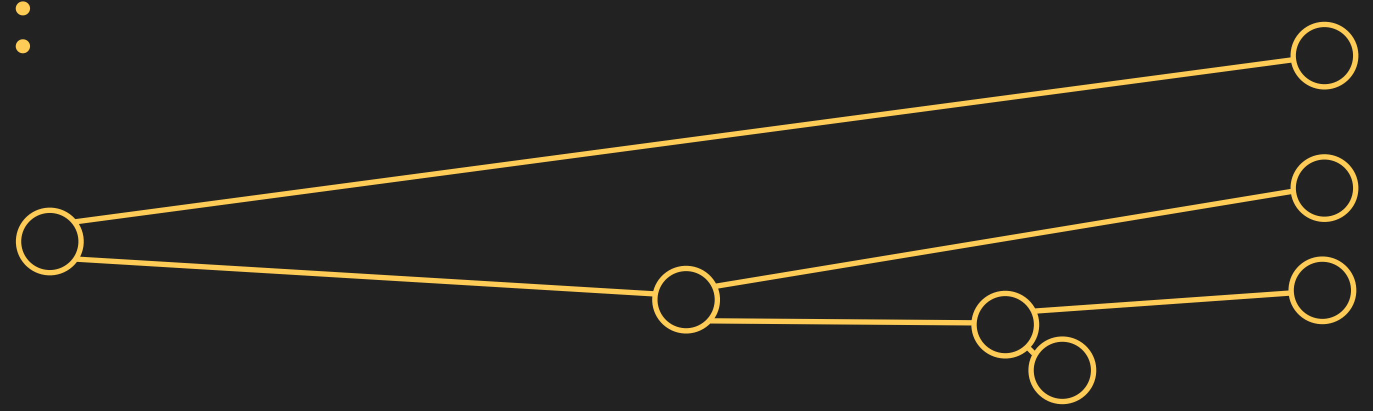
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:

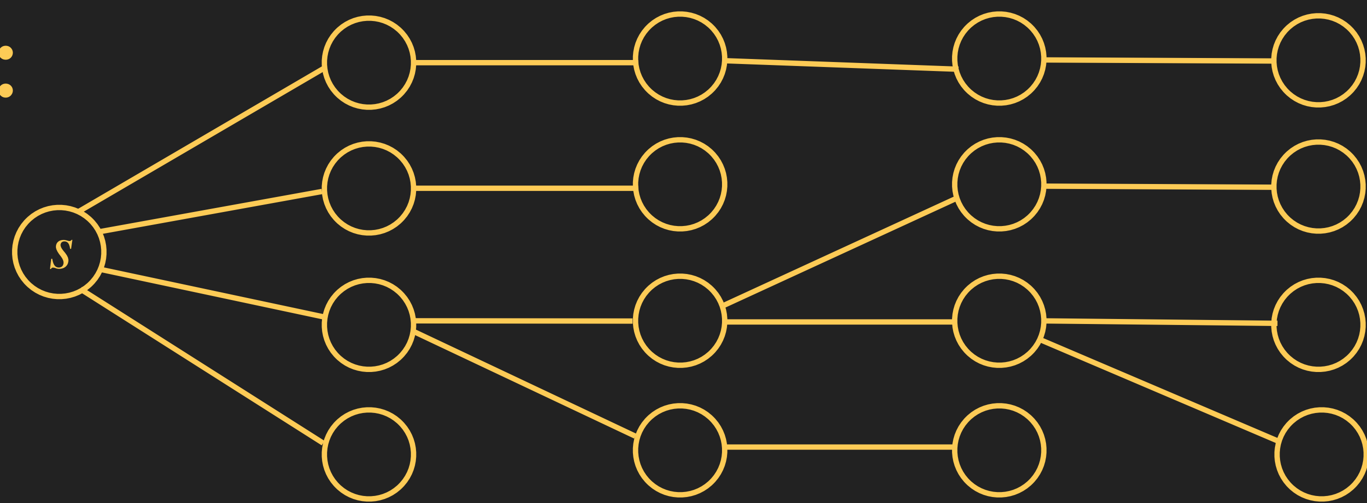


Reduction: LGT $\leq$ ETG

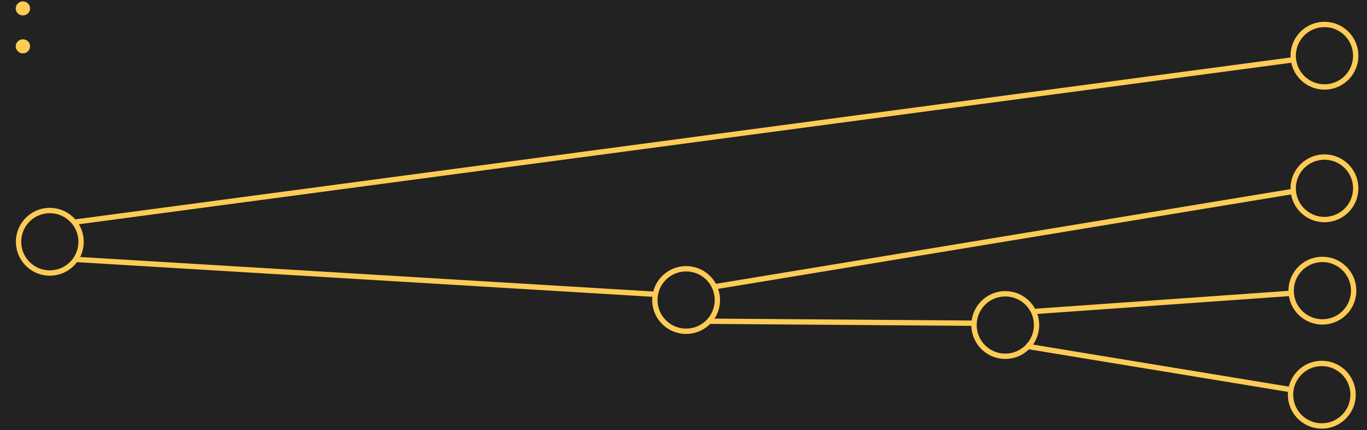
- Wlog layered graph is a tree [Fiat et al. '91]

  (build online the tree of shortest paths from s)

LGT:



ETG:



Observation: depth $\leq$ #leaves $= |L_i| \leq k$

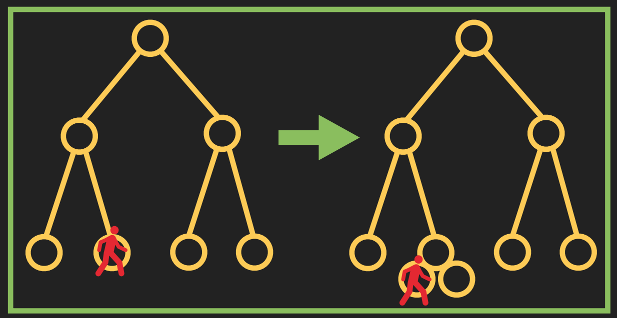
Algorithm

Algorithm

- Fork: Do nothing (almost)
- Growth:
- Deletion:

Algorithm

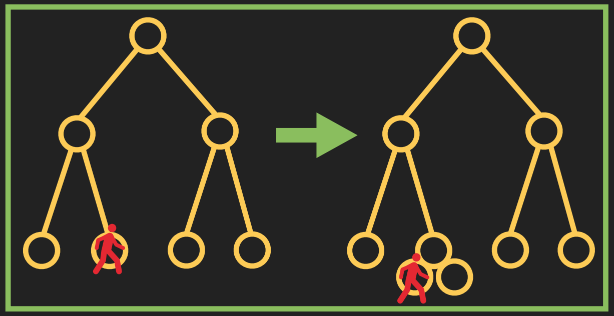
- Fork: Do nothing (almost)



- Growth:
- Deletion:

Algorithm

- Fork: Do nothing (almost)



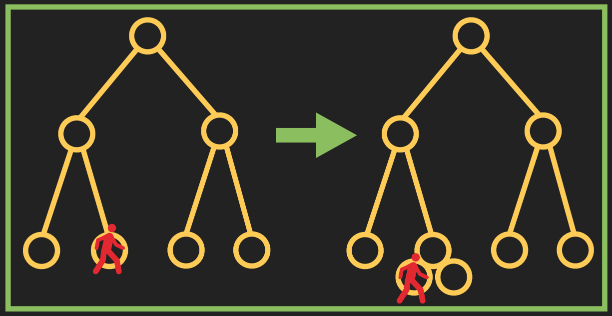
- Growth:

$$x'_u = -\frac{2x_u}{\tilde{w}_u}\tilde{w}'_u + \frac{x_u + \delta_u}{\tilde{w}_u}\left(\lambda_{p(u)} - \lambda_u\right)$$

- Deletion:

Algorithm

- Fork: Do nothing (almost)



- Growth:

x_u = probability mass in subtree of u

$\delta_u = \frac{1}{2^{\text{depth}(u)}}$

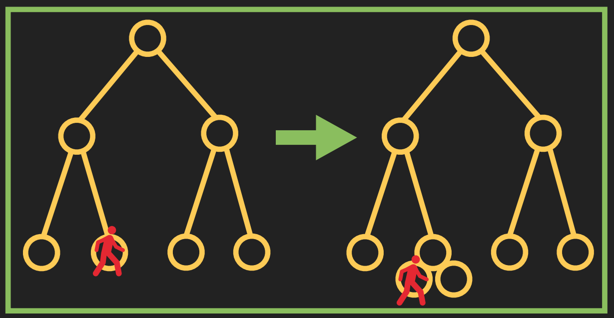
$$x'_u = -\frac{2x_u}{\tilde{w}_u}\tilde{w}'_u + \frac{x_u + \delta_u}{\tilde{w}_u}\underbrace{\left(\lambda_{p(u)} - \lambda_u\right)}_{\text{chosen s.t. x is well-defined probability}}$$

$\tilde{w}_u = \frac{2k-1}{2k - \text{depth(u)}} \cdot$ weight of edge $(u, p(u))$

- Deletion:

Algorithm

- Fork: Do nothing (almost)



- Growth:

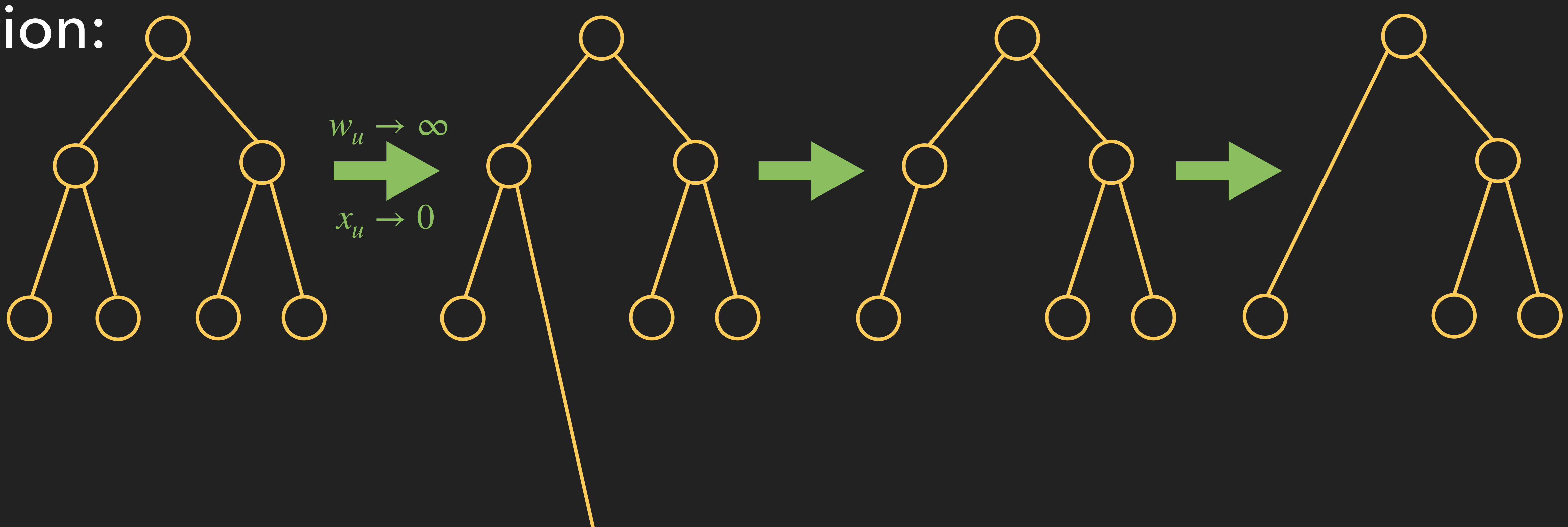
x_u = probability mass in subtree of u

$\delta_u = \frac{1}{2^{\text{depth}(u)}}$

$$x'_u = -\frac{2x_u}{\tilde{w}_u}\tilde{w}'_u + \frac{x_u + \delta_u}{\tilde{w}_u}\underbrace{\left(\lambda_{p(u)} - \lambda_u\right)}_{\text{chosen s.t. x is well-defined probability}}$$

$\tilde{w}_u = \frac{2k-1}{2k - \text{depth(u)}} \cdot$ weight of edge $(u, p(u))$

- Deletion:



Analysis

For a suitable potential function P, for any step (discrete or continuous) we have

$$\Delta\text{cost} + \Delta P \leq O(k^2) \cdot \Delta\text{opt}$$

Analysis

For a $\boxed{\text{suitable potential function } P,}$ for any step (discrete or continuous) we have

$$\Delta\text{cost} + \Delta P \leq O(k^2) \cdot \Delta\text{opt}$$

$$P := 2\sum_u \tilde{w}_u \left(4k y_u \log \frac{1+\delta_u}{x_u+\delta_u} + (2k - \text{depth}(u)) x_u \right)$$

Analysis

For a suitable potential function P, for any step (discrete or continuous) we have

$$\Delta\text{cost} + \Delta P \leq O(k^2) \cdot \Delta\text{opt}$$

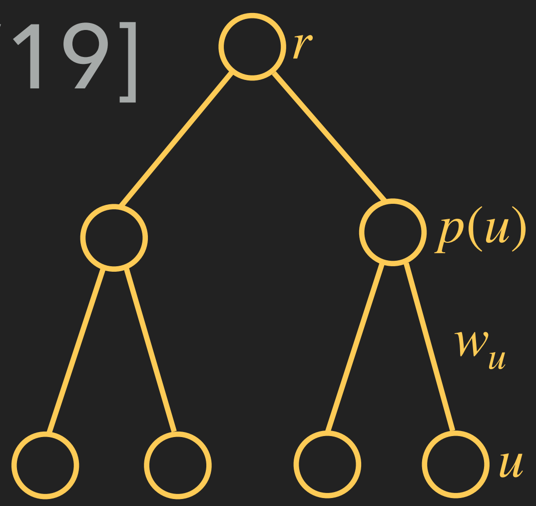
$$P := 2\sum_u \tilde{w}_u \left(4k y_u \log \frac{1+\delta_u}{x_u+\delta_u} + (2k - \text{depth}(u))x_u \right)$$

WHERE IS THIS ALL COMING FROM???

Metrical Task Systems (MTS) [Borodin, Linial, Saks 1987]

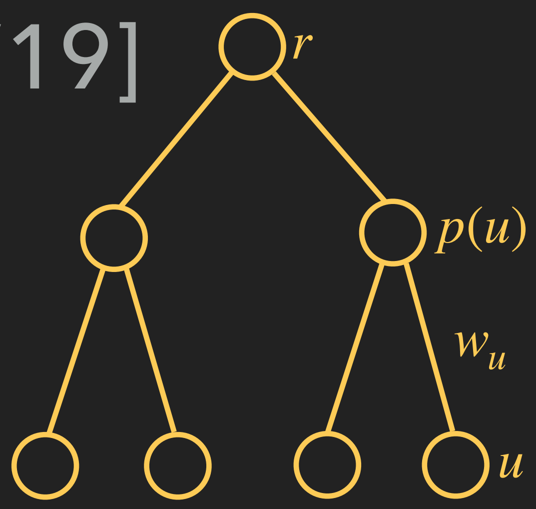
- metric space (M, d), $|M| = n$
- At time $t = 1,2,\ldots$
  - $c_t\colon M \to \mathbb{R}_+ \cup \{\infty\}$ revealed
  - Choose server position $p_t \in M$
  - Pay $d(p_{t-1}, p_t) + c_t(p_t)$

Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]



$$K := \left\{ x \in [0,1]^V \,\middle|\, x_r = 1, \forall u \neq \text{leaf}: x_u = \sum_{v:\, p(v)=u} x_v \right\}$$

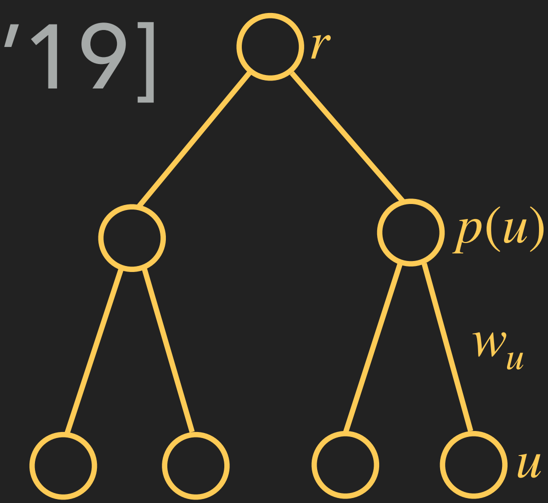
Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]



$$K := \left\{ x \in [0,1]^V \,\middle|\, x_r = 1, \forall u \neq \text{leaf}: x_u = \sum_{v:\, p(v)=u} x_v \right\}$$

- Cost vectors $c(t) \in \mathbb{R}_+^V$ (supported on leaves) appear in continuous time
- Algo maintains $x(t) \in K$
- Pays $\int \Big(\langle c(t), x(t) \rangle + \langle w, |\, x'(t) \,| \rangle \Big)\, dt$

Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]



$$K := \left\{ x \in [0,1]^V \;\middle|\; x_r = 1, \forall u \neq \text{leaf}: x_u = \sum_{v:\, p(v)=u} x_v \right\}$$

- Cost vectors $c(t) \in \mathbb{R}_+^V$ (supported on leaves) appear in continuous time
- Algo maintains $x(t) \in K$
- Pays $\int \big(\langle c(t), x(t) \rangle + \langle w, |x'(t)| \rangle \big)\, dt$

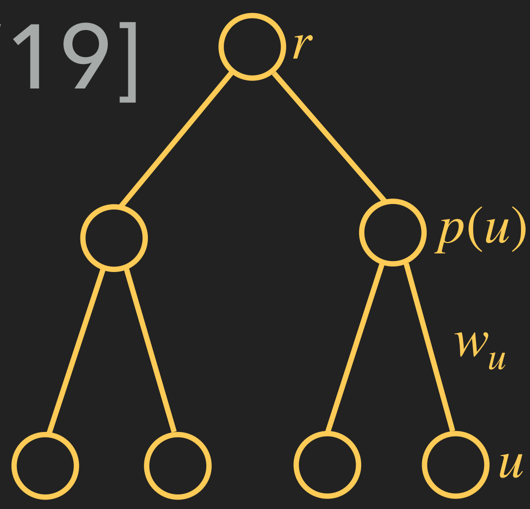
Algorithm:

$$x(t) = \arg\min_{x \in K} \text{opt}_t(x) + \Phi(x)$$

$\approx$ −weighted entropy

$\approx$ predictability of x

Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]



$$K := \left\{ x \in [0,1]^V \;\middle|\; x_r = 1, \forall u \neq \text{leaf}: x_u = \sum_{v:\, p(v)=u} x_v \right\}$$

- Cost vectors $c(t) \in \mathbb{R}_+^V$ (supported on leaves) appear in continuous time
- Algo maintains $x(t) \in K$
- Pays $\int \big(\langle c(t), x(t) \rangle + \langle w, |x'(t)| \rangle \big)\, dt$

Algorithm:

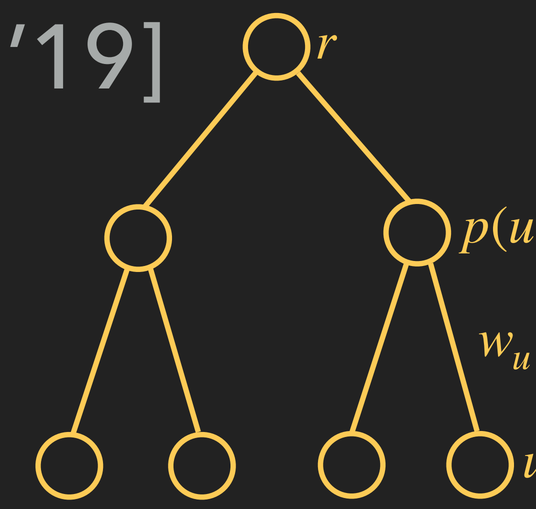
$$x(t) = \arg\min_{x \in K} \text{opt}_t(x) + \Phi(x)$$

$\approx$ $-$weighted entropy

$\approx$ predictability of x

General technique

Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]



$$K := \left\{ x \in [0,1]^V \;\middle|\; x_r = 1, \forall u \neq \text{leaf}: x_u = \sum_{v:\, p(v)=u} x_v \right\}$$

Algorithm:

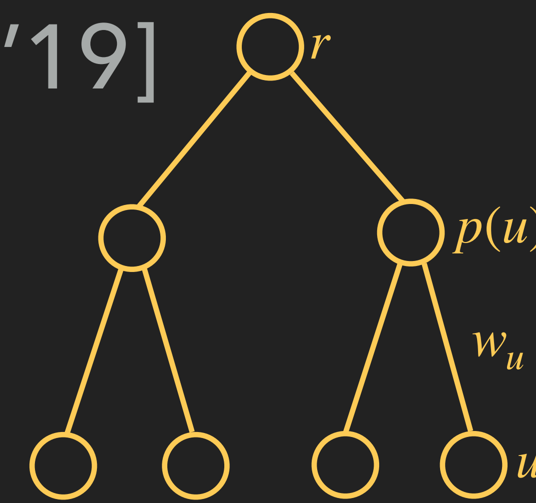
$$x(t) = \arg\min_{x \in K} \mathsf{opt}_t(x) + \Phi(x)$$

$\approx$ $-$weighted entropy

$\approx$ predictability of x

General technique

Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]



$$K := \left\{ x \in [0,1]^V \,\middle|\, x_r = 1, \forall u \neq \text{leaf}: x_u = \sum_{v:\, p(v)=u} x_v \right\}$$

Algorithm:

$$x(t) = \arg\min_{x \in K} \text{opt}_t(x) + \Phi(x)$$

$\approx$ −weighted entropy

$\approx$ predictability of x

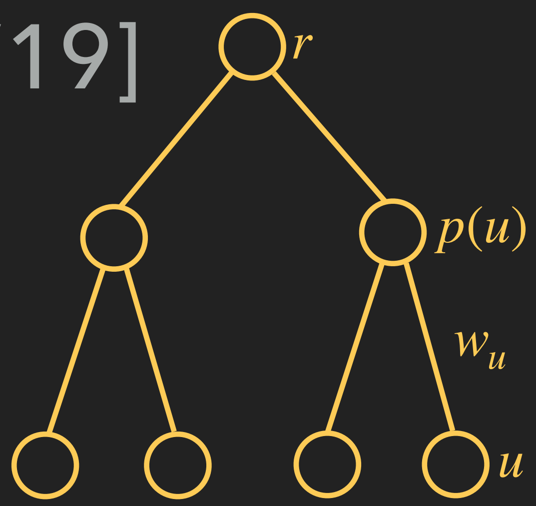
General technique

One can show, this is equivalent to:

$$\nabla^2 \Phi(x(t)) x'(t) \in -c(t) - N_K(x(t))$$

normal cone of K at $x(t)$

Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]



$$K := \left\{ x \in [0,1]^V \,\middle|\, x_r = 1, \forall u \neq \text{leaf}: x_u = \sum_{v:\, p(v)=u} x_v \right\}$$

Algorithm:

$$x(t) = \arg\min_{x \in K} \mathrm{opt}_t(x) + \Phi(x)$$

$\approx$ −weighted entropy

$\approx$ predictability of x

General technique

One can show, this is equivalent to:

$$\nabla^2 \Phi(x(t)) x'(t) \in -c(t) - N_K(x(t))$$

normal cone of K at $x(t)$

Mirror descent

Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]

- Let $y(t) \in K$ be offline algo
- $D(t) := \Phi(y(t)) - \Phi(x(t)) - \langle \nabla \Phi(x(t)), y(t) - x(t) \rangle$

Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]

- Let $y(t) \in K$ be offline algo
- $D(t) := \Phi(y(t)) - \Phi(x(t)) - \langle \nabla \Phi(x(t)), y(t) - x(t) \rangle$
- Short calculation $\Longrightarrow$

$$\langle c(t), x(t) \rangle + D'(t) \leq \langle c(t), y(t) \rangle + \mathrm{Lip}_\Phi \cdot \langle w, |y'(t)| \rangle$$

Randomized MTS on Trees [Bubeck,Cohen,Lee,Lee '19]

- Let $y(t) \in K$ be offline algo
- $D(t) := \Phi(y(t)) - \Phi(x(t)) - \langle \nabla \Phi(x(t)), y(t) - x(t) \rangle$
- Short calculation $\implies$

$$\langle c(t), x(t) \rangle + D'(t) \leq \langle c(t), y(t) \rangle + \mathrm{Lip}_\Phi \cdot \langle w, |y'(t)| \rangle$$

Choosing $\Phi(x) = \sum_u w_u (x_u + \delta_u) \log(x_u + \delta_u)$ for fixed $\delta \in K$

[Bubeck,Cohen,Lee,Lee '19] show

$$\langle w, (x'(t))_+ \rangle + \Psi'(t) \leq \mathrm{depth} \cdot \langle c(t), x(t) + \delta \rangle$$

where $\Psi(t) := -\sum_u \mathrm{depth}(u) \cdot w_u x_u(t)$

Algorithm for Evolving Tree Game

Idea: Same algo, but with $c(t) := w'(t)$

Algorithm for Evolving Tree Game

Idea: Same algo, but with $c(t) := w'(t)$

Three complications:

Algorithm for Evolving Tree Game

Idea: Same algo, but with $c(t) := w'(t)$

Three complications:

$\boxed{\text{Tree topology (and hence } K\text{) evolves} \rightsquigarrow \text{cannot choose fixed } \delta \in K}$

Algorithm for Evolving Tree Game

Idea: Same algo, but with $c(t) := w'(t)$

Three complications:

$\boxed{\text{Tree topology (and hence } K\text{) evolves} \rightsquigarrow \text{cannot choose fixed } \delta \in K}$

$\boxed{\text{When depth}(u) \text{ decreases, } \Psi \text{ increases}}$

Algorithm for Evolving Tree Game

Idea: Same algo, but with $c(t) := w'(t)$

Three complications:

Tree topology (and hence K) evolves $\rightsquigarrow$ cannot choose fixed $\delta \in K$

When depth(u) decreases, Ψ increases

When w grows, D increases

$\boxed{\text{Tree topology (and hence } K\text{) evolves} \rightsquigarrow \text{cannot choose fixed } \delta \in K}$

$\boxed{\text{Tree topology (and hence } K\text{) evolves} \rightsquigarrow \text{cannot choose fixed } \delta \in K}$

In MTS, choice $\delta_u = \dfrac{\text{\# leaves below } u}{\text{\# leaves}}$ ensures

$$\mathrm{Lip}_\Phi = O\left(\log \frac{1}{\min_u \delta_u}\right) = O(\log(\text{\# leaves}))$$

This δ_u can increase/decrease due to Fork/Delete

$\implies$ bad effects on D

$\boxed{\text{Tree topology (and hence } K\text{) evolves} \rightsquigarrow \text{cannot choose fixed } \delta \in K}$

In MTS, choice $\delta_u = \dfrac{\text{\# leaves below } u}{\text{\# leaves}}$ ensures

$$\mathrm{Lip}_\Phi = O\left(\log \frac{1}{\min_u \delta_u}\right) = O(\log(\text{\# leaves}))$$

This δ_u can increase/decrease due to Fork/Delete

$\implies$ bad effects on D

Solution: $\delta_u := 2^{-\mathrm{depth}(u)}$

Then $\delta \in K$ and δ_u only increases $\implies$ good effects on D

$$\mathrm{Lip}_\Phi = O\left(\log \frac{1}{\min_u \delta_u}\right) = O(\mathrm{depth})$$

Algorithm for Evolving Tree Game

Three complications:

$\boxed{\text{Tree topology (and hence } K\text{) evolves} \rightsquigarrow \text{cannot choose fixed } \delta \in K}$

$\boxed{\text{When depth}(u) \text{ decreases, } \Psi \text{ increases}}$

$\boxed{\text{When } w \text{ grows, } D \text{ increases}}$

When depth(u) decreases, Ψ increases

When depth(u) decreases, Ψ increases

Solution: Replace w_u by $\tilde{w}_u$ to cancel bad effects.

When depth(u) decreases, Ψ increases

Solution: Replace w_u by $\tilde{w}_u$ to cancel bad effects.

Specifically, $\tilde{w}_u := \dfrac{2k-1}{2k-\text{depth}(u)} w_u$

Then $w_u \le \tilde{w}_u \le 2w_u$, so error is small

Algorithm for Evolving Tree Game

Three complications:

Tree topology (and hence K) evolves $\rightsquigarrow$ cannot choose fixed $\delta \in K$

When depth(u) decreases, Ψ increases

When w grows, D increases

When w grows, D increases

When w grows, D increases

Solution: Essentially, replace

$$D = \sum_u \tilde{w}_u \left((y_u + \delta_u) \log \frac{y_u + \delta_u}{x_u + \delta_u} + x_u - y_u \right) \text{ and } c(t) = w'(t)$$

by

$$D = \sum_u \tilde{w}_u \left(2y_u \log \frac{y_u + \delta_u}{x_u + \delta_u} + x_u - y_u \right) \text{ and } c(t) = \frac{2x}{x + \delta} w'(t)$$

Now increase of D due to growth of w can be charged to OPT.

Algorithm for Evolving Tree Game

Tree topology (and hence K) evolves $\rightsquigarrow$ cannot choose fixed $\delta \in K$

When depth(u) decreases, Ψ increases

When w grows, D increases

- Mirror descent works even in evolving metric spaces
- $O(\text{depth} \cdot \log n)$-competitiveness for MTS becomes:
  - $O(\text{depth}^2)$ for evolving tree game
  - $O(k^2)$ for LGT and chasing small sets

The Randomized k-Server Conjecture is *****!

JOINT WORK WITH

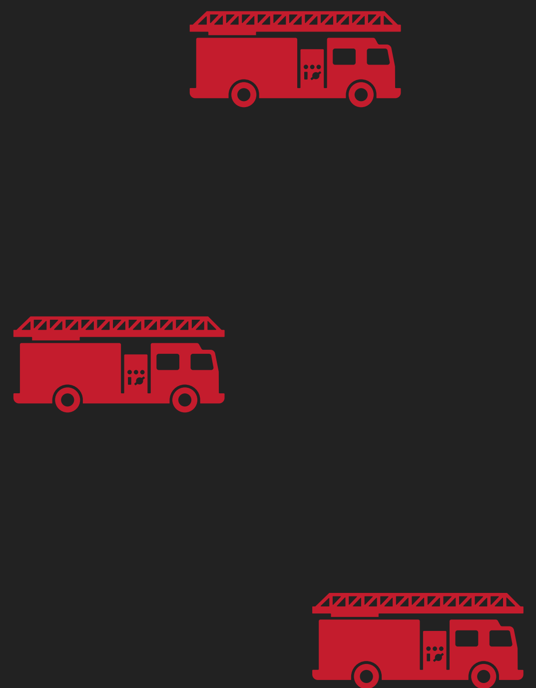
SÉBASTIEN BUBECK (MICROSOFT RESEARCH)
YUVAL RABANI (HEBREW UNIVERSITY OF JERUSALEM)

k-server problem

- k servers in metric space (M, d)
- At time $t = 1, 2, \ldots$
  - point $r_t \in M$ requested
  - A server must move to r_t

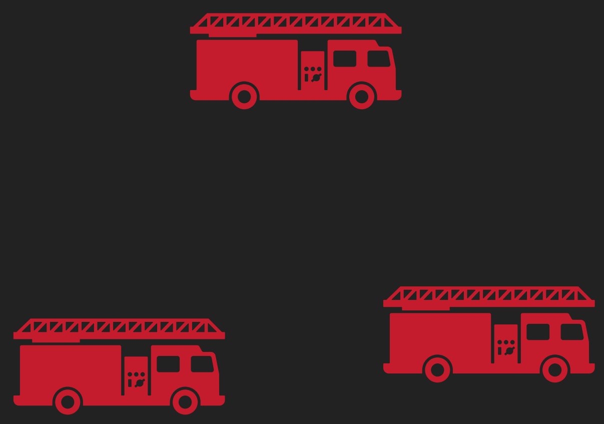
k-server problem

- k servers in metric space (M, d)
- At time $t = 1,2,\ldots$
    - point $r_t \in M$ requested
    - A server must move to r_t



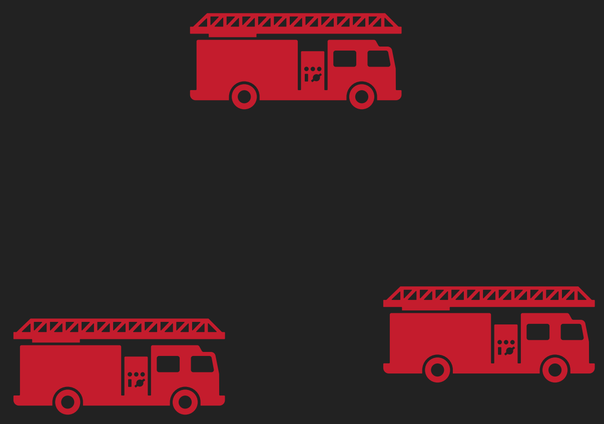
k-server problem

- k servers in metric space (M, d)
- At time $t = 1,2,\ldots$
  - point $r_t \in M$ requested
  - A server must move to r_t



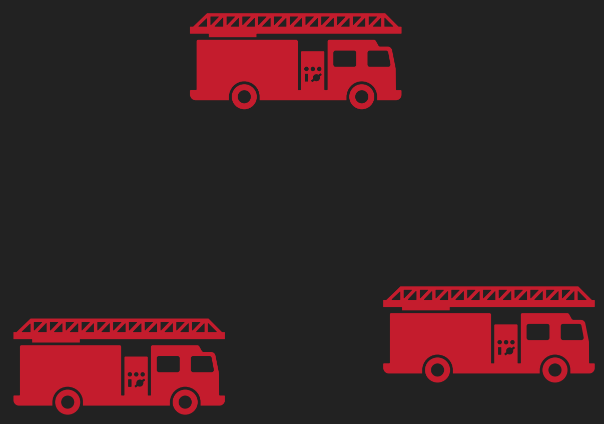
k-server problem

- k servers in metric space (M, d)
- At time $t = 1,2,\ldots$
  - point $r_t \in M$ requested
  - A server must move to r_t
- Cost = distance traveled



k-server problem

- k servers in metric space (M, d)
- At time $t = 1,2,\ldots$
  - point $r_t \in M$ requested
  - A server must move to r_t
- Cost = distance traveled



k-server often called "holy grail of competitive analysis"

k-server conjecture: $\exists$ k-competitive deterministic algorithm

k-server conjecture: $\exists\ k$-competitive deterministic algorithm

State of the art:

- $\geq k$ [Manasse,McGeoch,Sleator 88]
- $\leq 2k - 1$ [Koutsoupias,Papadimitriou 94]
- $= k$ in special cases

k-server conjecture: $\exists$ k-competitive deterministic algorithm

State of the art:

- $\geq k$ [Manasse,McGeoch,Sleator 88]
- $\leq 2k - 1$ [Koutsoupias,Papadimitriou 94]
- $= k$ in special cases

Randomized k-server conjecture: $\exists$ $O(\log k)$-comp. rand. algo

k-server conjecture: $\exists$ k-competitive deterministic algorithm

State of the art:

- $\geq k$ [Manasse,McGeoch,Sleator 88]
- $\leq 2k - 1$ [Koutsoupias,Papadimitriou 94]
- $= k$ in special cases

Randomized k-server conjecture: $\exists$ $O(\log k)$-comp. rand. algo

State of the art:

- $\Omega(\log k \ / \ \log \log k)$ [Bartal,Bollobas,Mendel 01,
  Bartal,Linial,Mendel,Naor 03]
- $O(\log^2 k \log n)$ in n-point metrics [Bubeck,Cohen,Lee,Lee,Madry 18]
  $O(\log^3 k \log \Delta)$ where Δ=aspect ratio [Bubeck,Cohen,Lee,Lee,Madry 18]
- $\Theta(\log k)$ in special cases

Randomized k-server conjecture:

$\exists$ an $O(\log k)$-competitive randomized algorithm

Theorem [Bubeck,Coester,Rabani 23]:

$\exists$ no $O(\log k)$-competitive randomized algorithm

Theorem [Bubeck,Coester,Rabani 23]:
$\exists$ no $O(\log k)$-competitive randomized algorithm

More precisely:

Theorem: Comp. ratio is $\Omega(\log^2 k)$ in *some* metrics of $k+1$ points

Theorem [Bubeck,Coester,Rabani 23]:

$\exists$ no $O(\log k)$-competitive randomized algorithm

More precisely:

Theorem: Comp. ratio is $\Omega(\log^2 k)$ in *some* metrics of $k+1$ points

Also tight universal lower bound:

Theorem: Comp. ratio is $\Omega(\log k)$ in *all* metrics of $>k$ points

Metrical Task Systems (MTS)

- metric space (M, d), $|M| = n$
- 1 server, initially at $p_0 \in M$
- At time $t = 1,2,\ldots$
  - $c_t\colon M \to \mathbb{R}_+ \cup \{\infty\}$ revealed
  - Choose $p_t \in M$
  - Pay $d(p_{t-1}, p_t) + c_t(p_t)$

Metrical Task Systems (MTS)

- metric space (M, d), $|M| = n$
- 1 server, initially at $p_0 \in M$
- At time $t = 1,2,\ldots$
  - $c_t \colon M \to \mathbb{R}_+ \cup \{\infty\}$ revealed
  - Choose $p_t \in M$
  - Pay $d(p_{t-1}, p_t) + c_t(p_t)$

MTS with $c_t \colon M \to \{0,\infty\}$ $\equiv$ $(n-1)$-server problem

Metrical Task Systems (MTS)

- metric space (M, d), $|M| = n$
- 1 server, initially at $p_0 \in M$
- At time $t = 1, 2, \ldots$
  - $c_t : M \to \mathbb{R}_+ \cup \{\infty\}$ revealed
  - Choose $p_t \in M$
  - Pay $d(p_{t-1}, p_t) + c_t(p_t)$

MTS with $c_t : M \to \{0, \infty\}$ $\equiv$ $(n-1)$-server problem

Corollary: Comp. ratio of MTS is $\Omega(\log^2 n)$ in *some* metrics (tight)
$\Omega(\log n)$ in *all* metrics (tight)

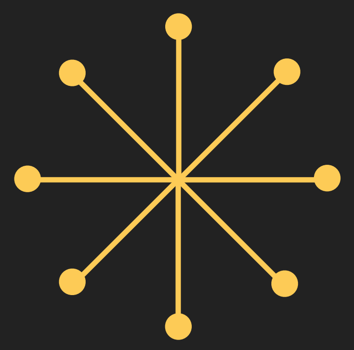
Previous best existential LB was $\Omega(\log n)$:

[Borodin,Linial,Saks 87]

Previous best existential LB was $\Omega(\log n)$:

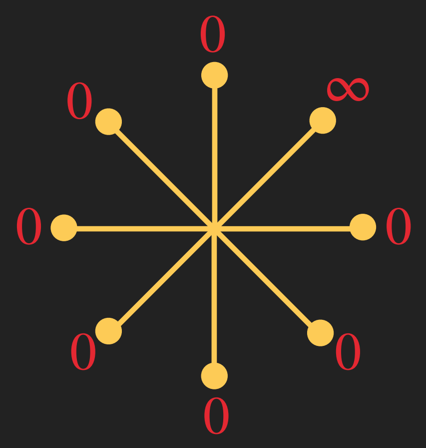
[Borodin,Linial,Saks 87]

- metric space (M, d) with $d(x, y) = 1_{x \neq y}$



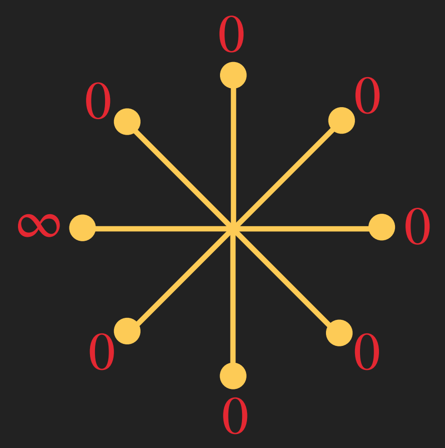
Previous best existential LB was $\Omega(\log n)$:

[Borodin,Linial,Saks 87]



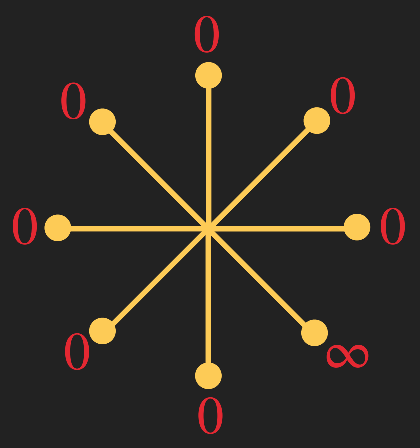
- metric space (M, d) with $d(x, y) = 1_{x \neq y}$
- $c_t(p) := \begin{cases} \infty & p = r_t \\ 0 & p \neq r_t \end{cases}$ for $r_t \in M$ unif. at random

Previous best existential LB was $\Omega(\log n)$:
[Borodin,Linial,Saks 87]



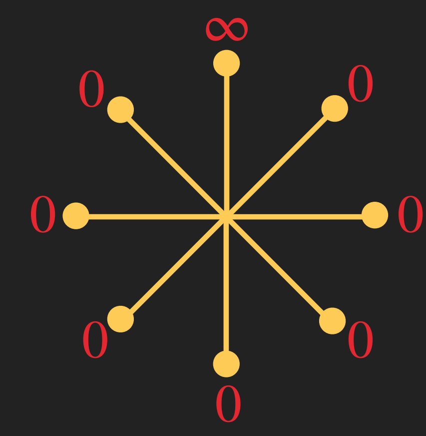
- metric space (M, d) with $d(x, y) = 1_{x \neq y}$
- $c_t(p) := \begin{cases} \infty & p = r_t \\ 0 & p \neq r_t \end{cases}$ for $r_t \in M$ unif. at random

Previous best existential LB was $\Omega(\log n)$:
[Borodin,Linial,Saks 87]



- metric space (M, d) with $d(x, y) = 1_{x \neq y}$
- $c_t(p) := \begin{cases} \infty & p = r_t \\ 0 & p \neq r_t \end{cases}$ for $r_t \in M$ unif. at random

Previous best existential LB was $\Omega(\log n)$:
[Borodin,Linial,Saks 87]



- metric space (M, d) with $d(x, y) = 1_{x \neq y}$
- $c_t(p) := \begin{cases} \infty & p = r_t \\ 0 & p \neq r_t \end{cases}$ for $r_t \in M$ unif. at random
  - $\mathbb{E}[\text{cost}] = \#\text{requests} / n$
  - $\mathbb{E}[\text{opt}] = \#\text{requests} / \Omega(n \log n)$

Some Intuition

The Road Not Taken [Robert Frost, 1915]

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

Some Intuition

The Road Not Taken [Robert Frost, 1915]

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

Some Intuition

The Road Not Taken [Robert Frost, 1915]

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

Some Intuition

The Road Not Taken [Robert Frost, 1915]

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

Some Intuition

The Road Not Taken [Robert Frost, 1915]

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

Some Intuition

The Road Not Taken [Robert Frost, 1915]

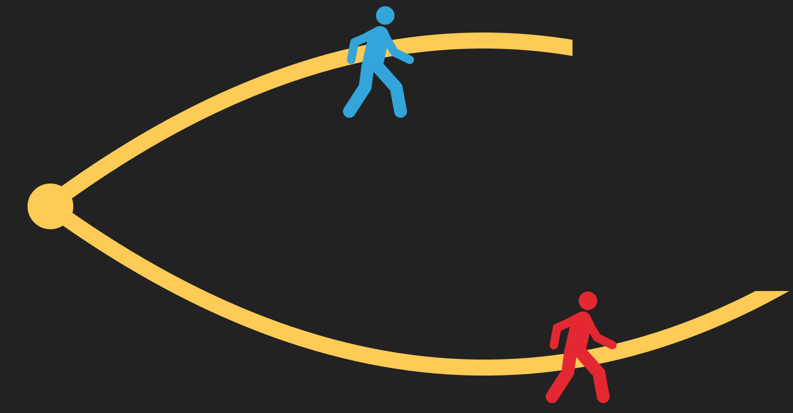
Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

Some Intuition



The Road Not Taken [Robert Frost, 1915]

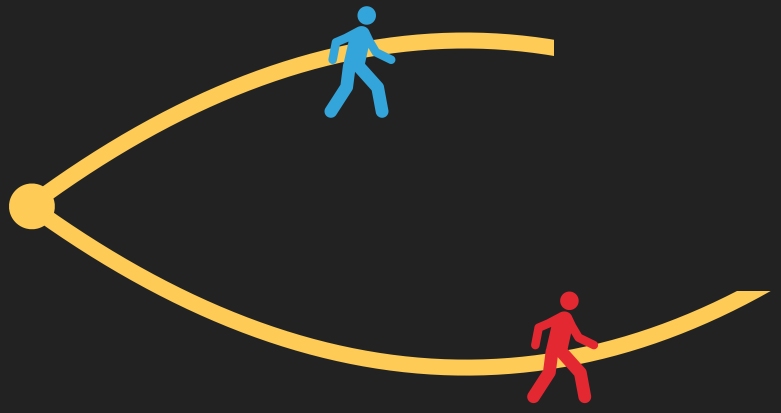
Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

Some Intuition



The Road Not Taken [Robert Frost, 1915]

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
Had worn them really about the same,

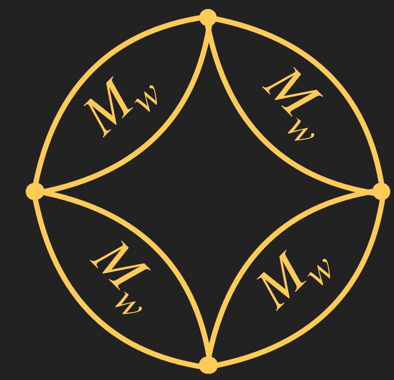
And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back.

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.

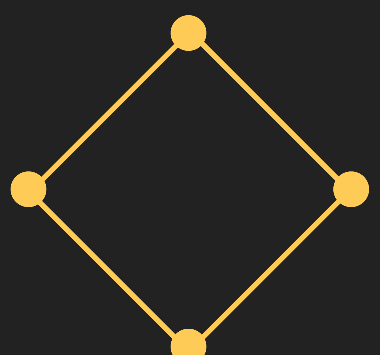
□

Metric space is similar to diamond graph

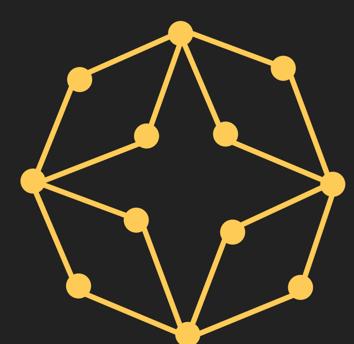
$$M_{w+1} =$$



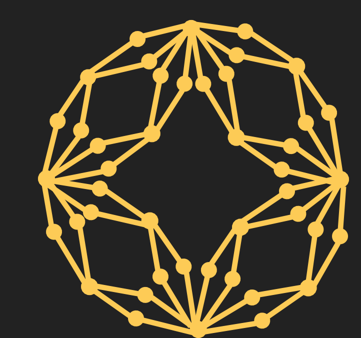
$$M_1 =$$

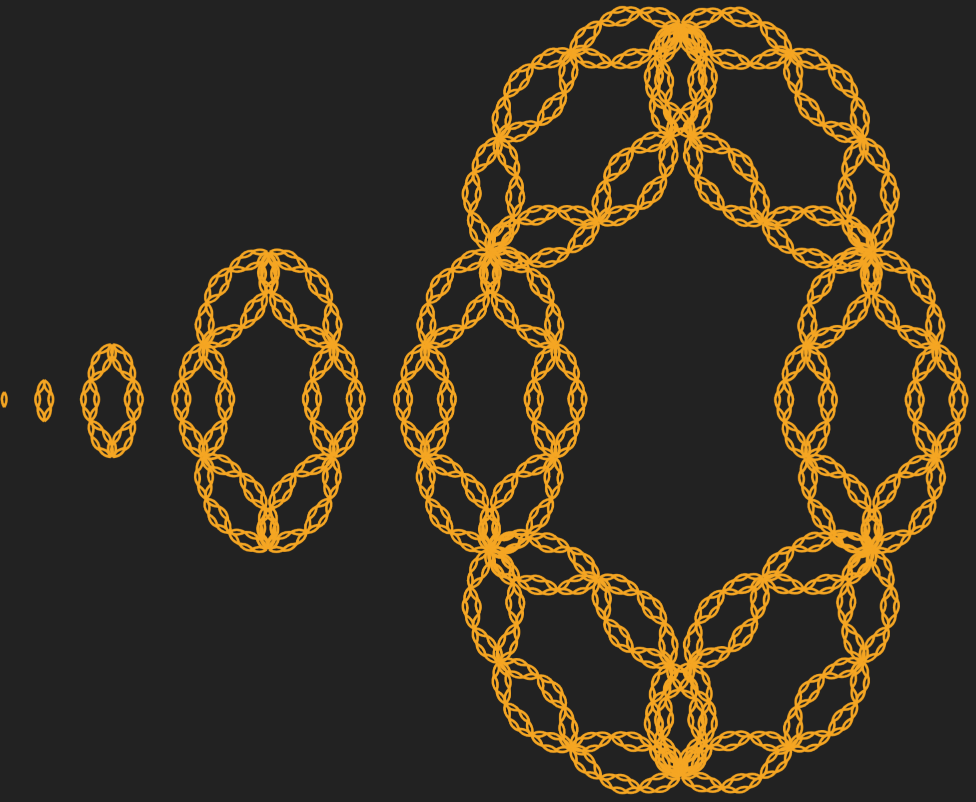


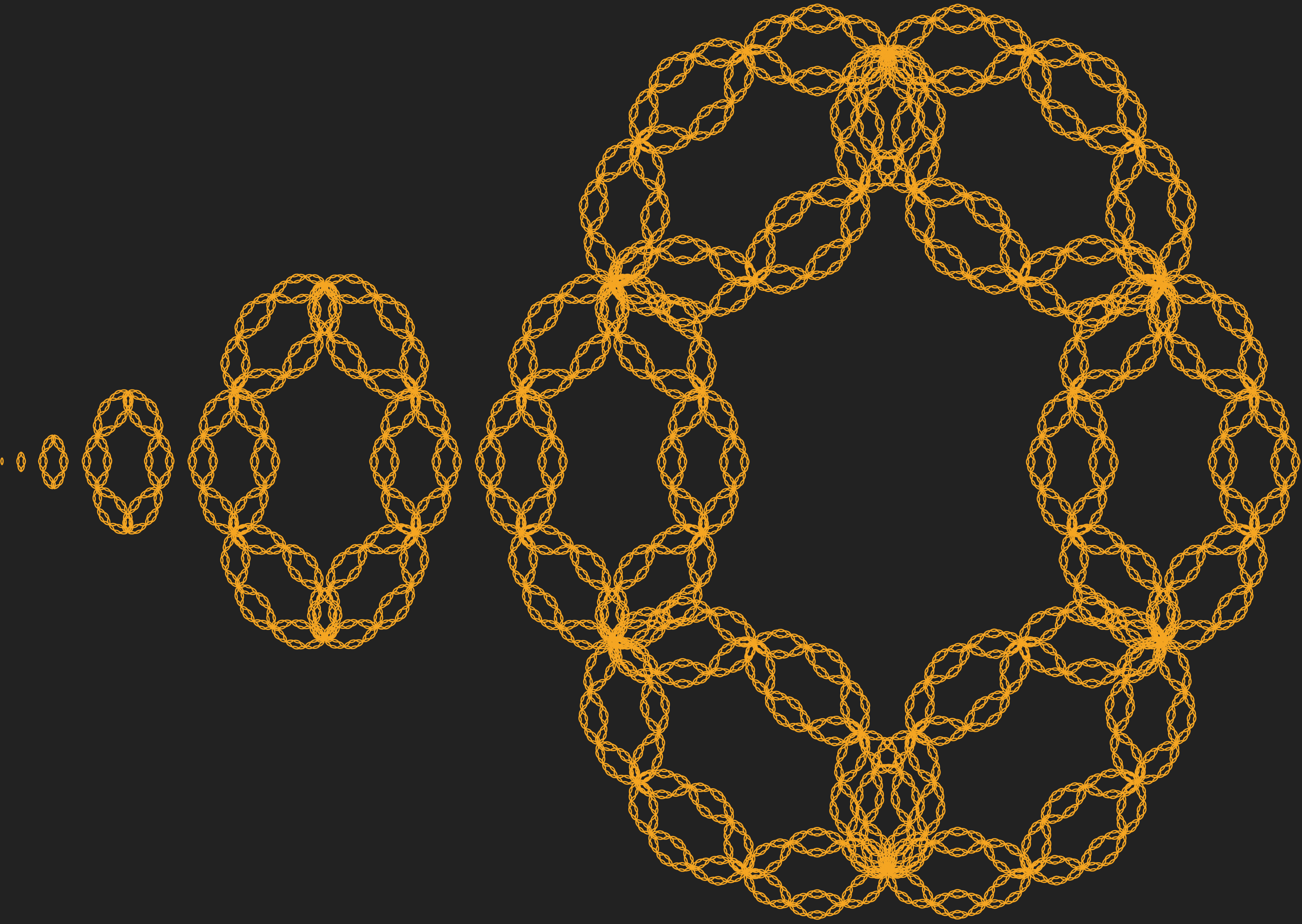
$$M_2 =$$



$$M_3 =$$







Proof of $\Omega\left(\left(\frac{\log n}{\log\log n}\right)^2\right)$

- Construct metric space 

recursively
- Goal: (random) sequence 

s.t. opt $=$ diam(M_w)

  cost $\geq R_w \cdot$ diam(M_w)

- Construct metric space 

recursively
- Goal: (random) sequence 

s.t. opt $= \mathrm{diam}(M_w)$

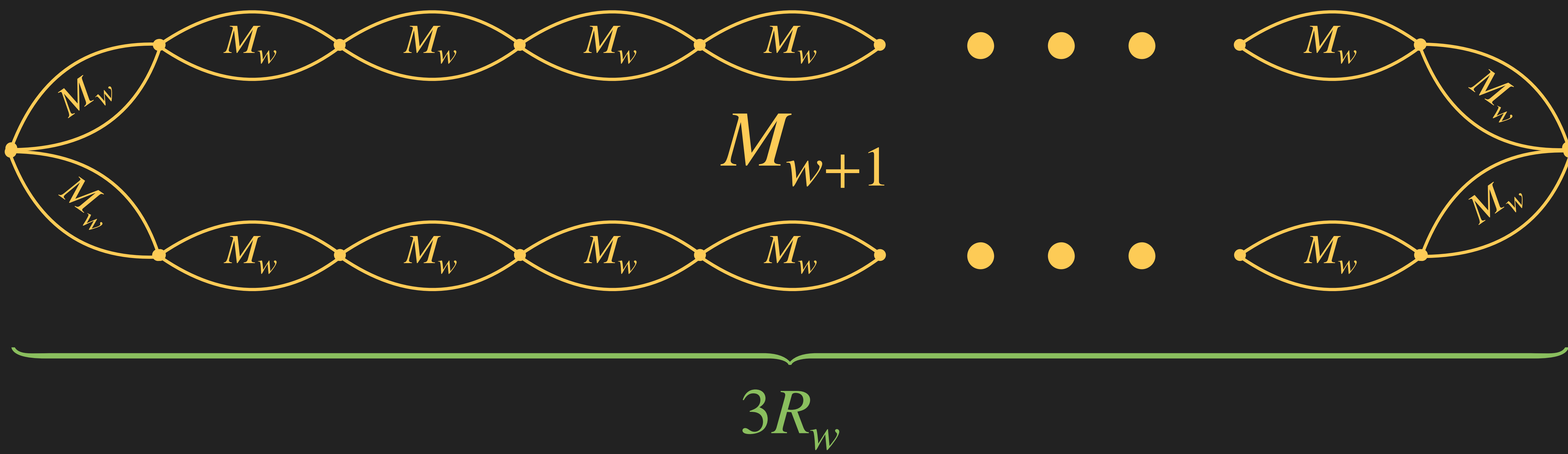
  cost $\geq R_w \cdot \mathrm{diam}(M_w)$

- Construct metric space 

recursively
- Goal: (random) sequence 

s.t. opt $= \mathrm{diam}(M_w)$

  cost $\geq R_w \cdot \mathrm{diam}(M_w)$

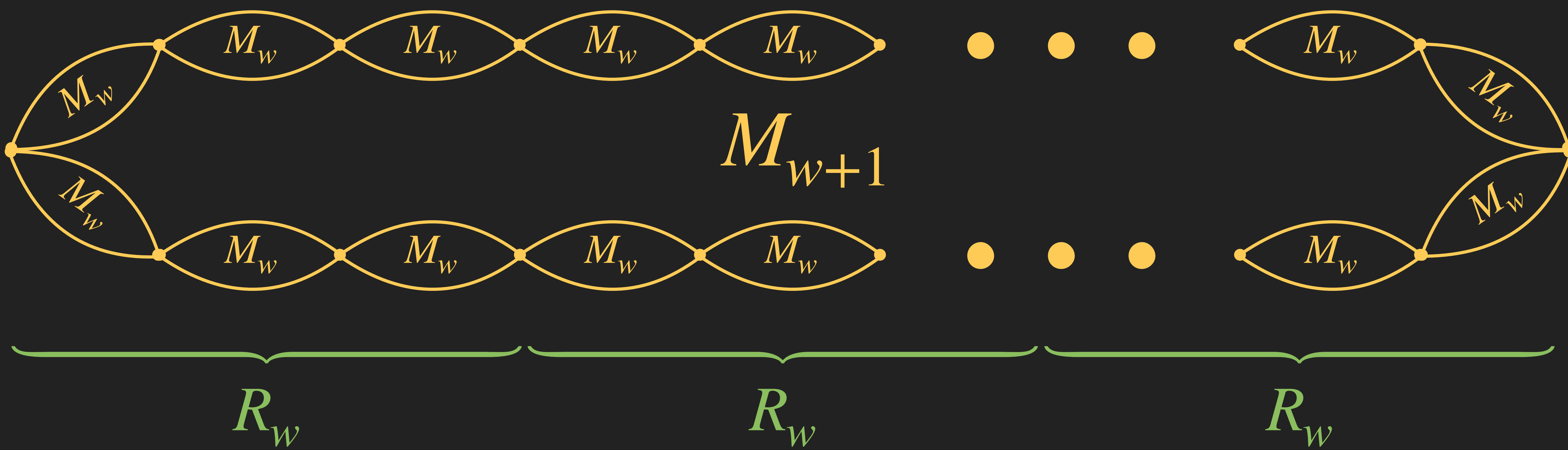


- Construct metric space 

recursively
- Goal: (random) sequence 

s. t. opt $= \mathrm{diam}(M_w)$

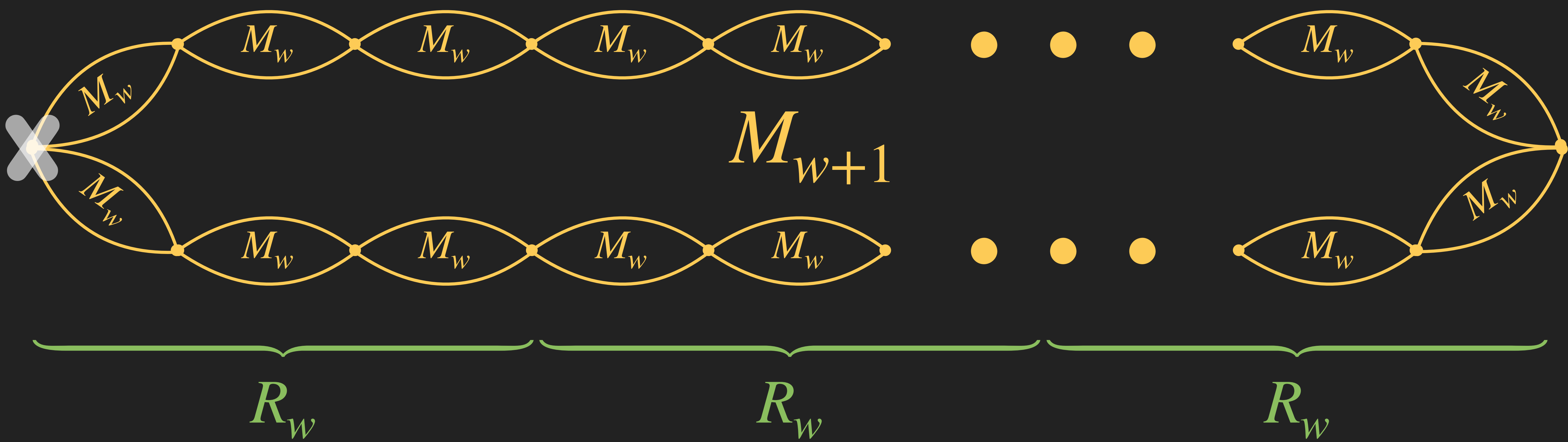
  cost $\geq R_w \cdot \mathrm{diam}(M_w)$



- Construct metric space  recursively

- Goal: (random) sequence  s. t. opt $=$ diam(M_w)

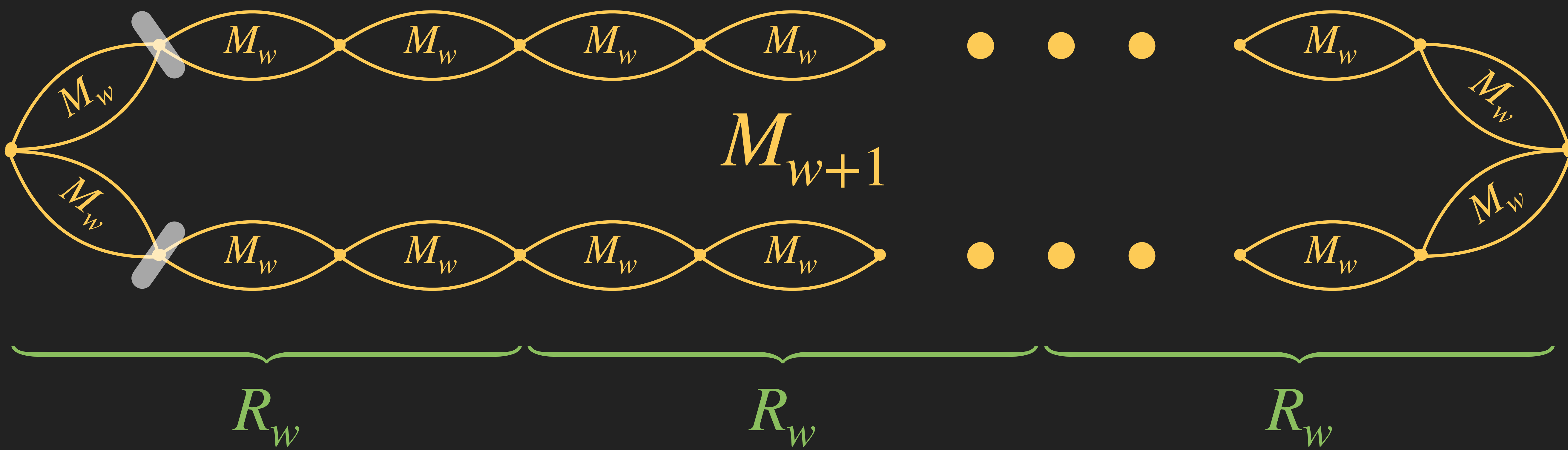
  cost $\geq R_w \cdot$ diam(M_w)



- Construct metric space 

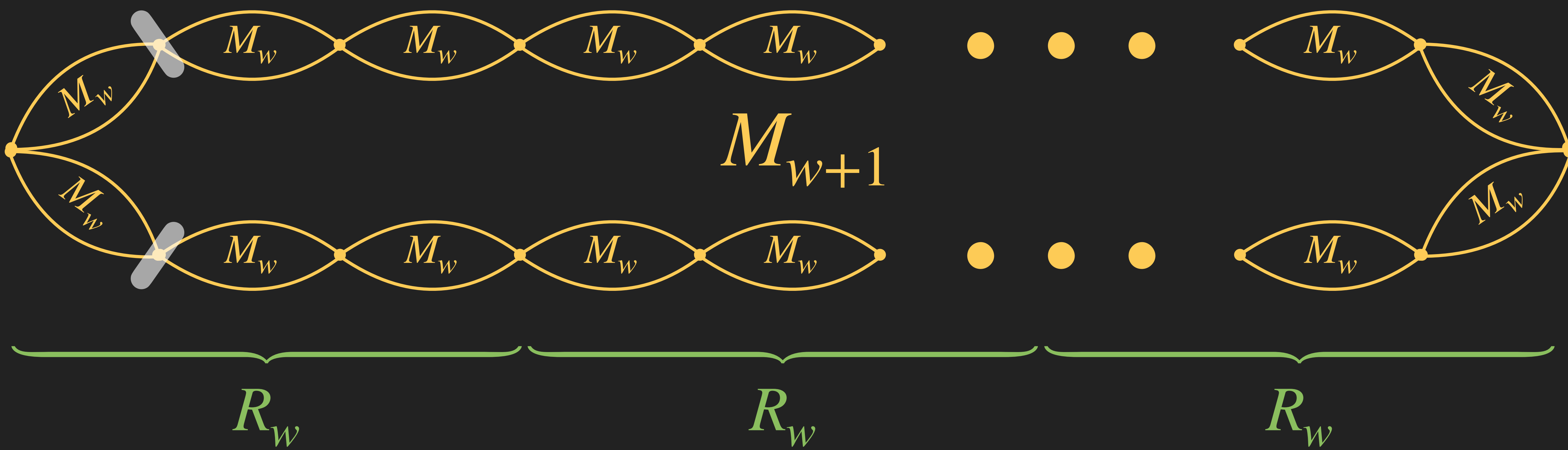
recursively
- Goal: (random) sequence 

s. t. opt $= \mathrm{diam}(M_w)$
cost $\geq R_w \cdot \mathrm{diam}(M_w)$



- Construct metric space M_w recursively
- Goal: (random) sequence M_w s. t. opt $= \mathrm{diam}(M_w)$

  cost $\geq R_w \cdot \mathrm{diam}(M_w)$



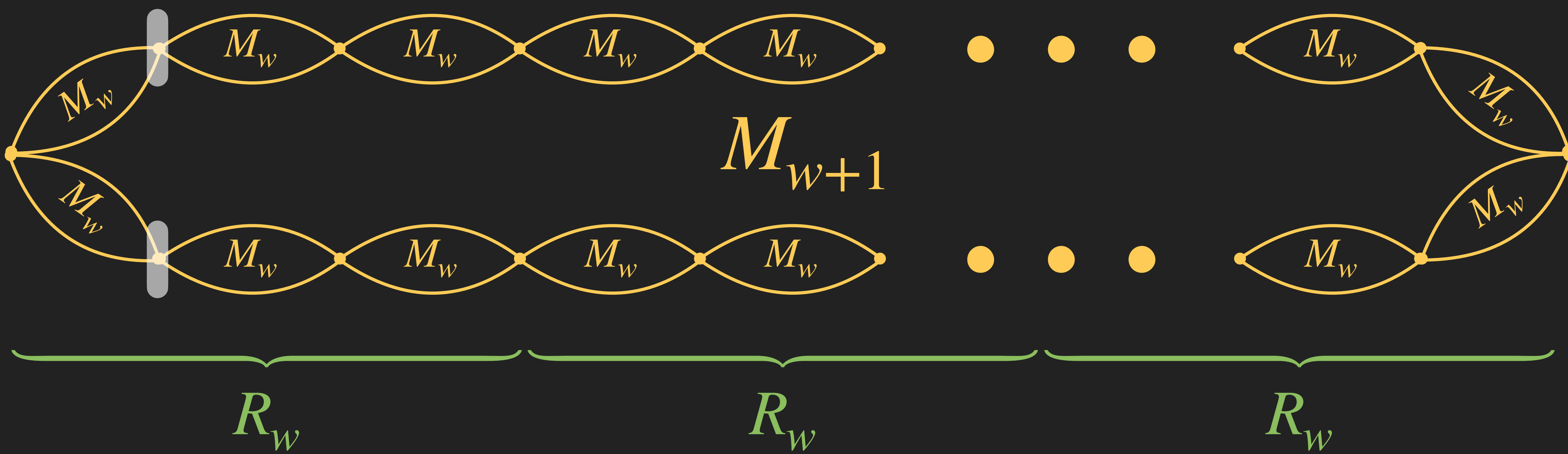
- Construct metric space 

recursively

- Goal: (random) sequence 

s. t. opt $=$ diam(M_w)

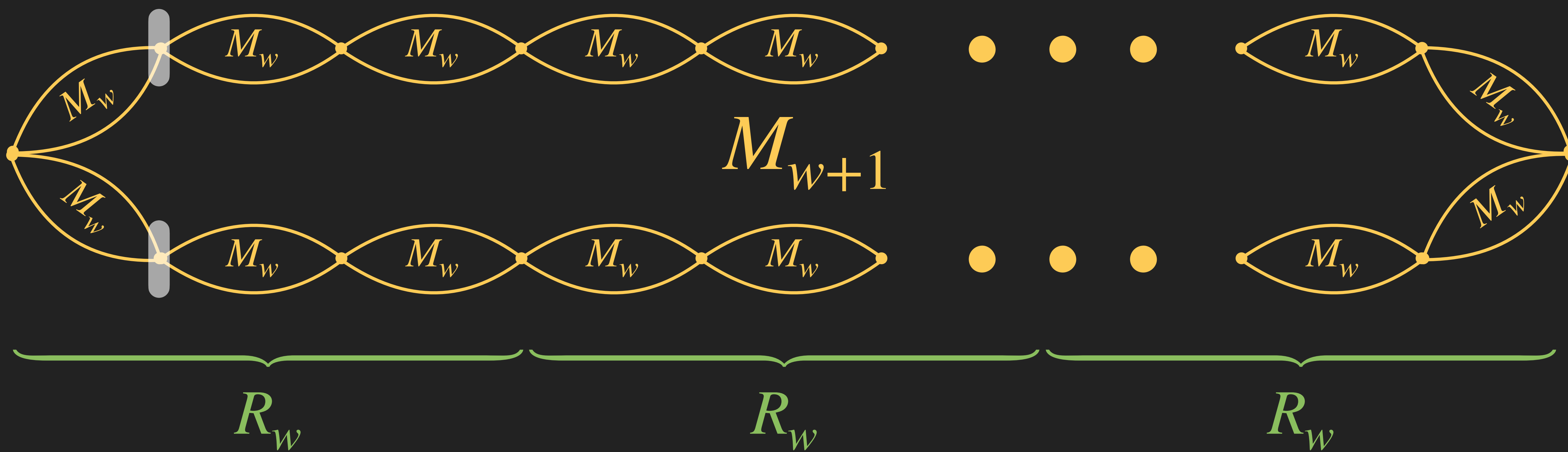
cost $\geq R_w \cdot$ diam(M_w)



- Construct metric space  recursively

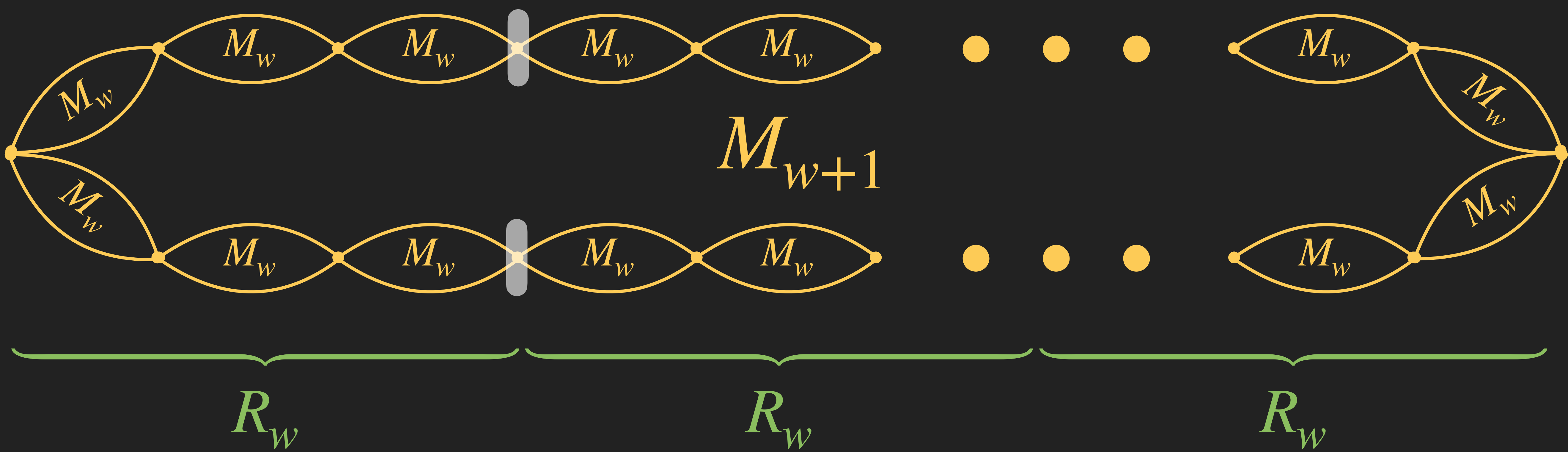
- Goal: (random) sequence  s. t. opt = diam(M_w)

cost $\geq R_w \cdot$ diam(M_w)



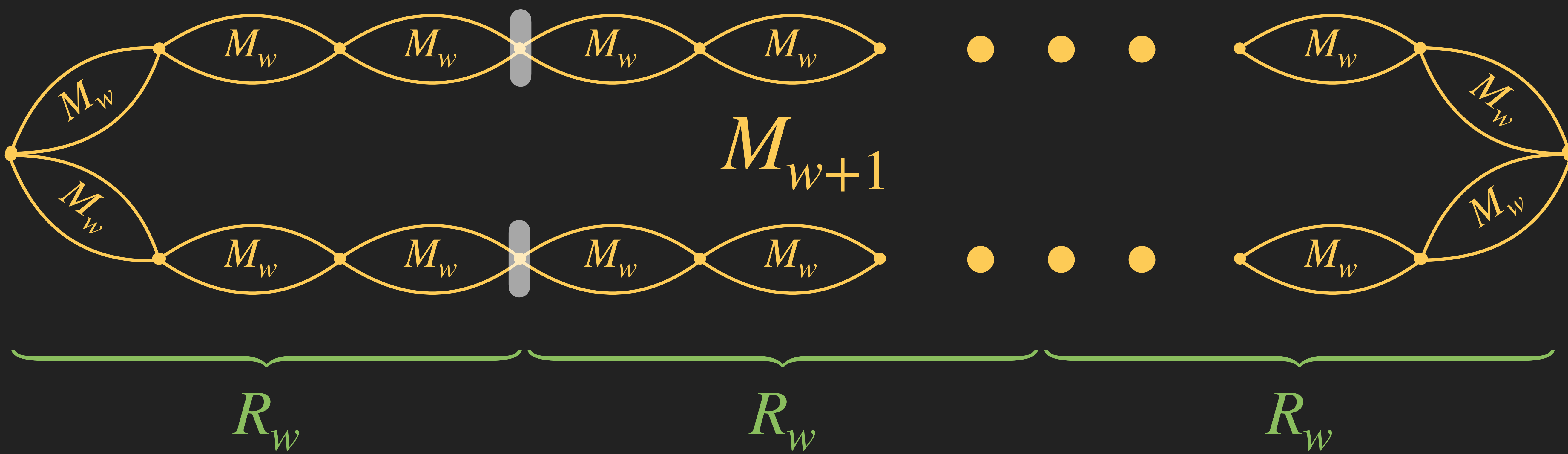
- Construct metric space M_w recursively
- Goal: (random) sequence M_w s. t. opt $= \mathrm{diam}(M_w)$

  cost $\geq R_w \cdot \mathrm{diam}(M_w)$



- Construct metric space  recursively
- Goal: (random) sequence  s. t. opt $= \mathrm{diam}(M_w)$

  cost $\geq R_w \cdot \mathrm{diam}(M_w)$



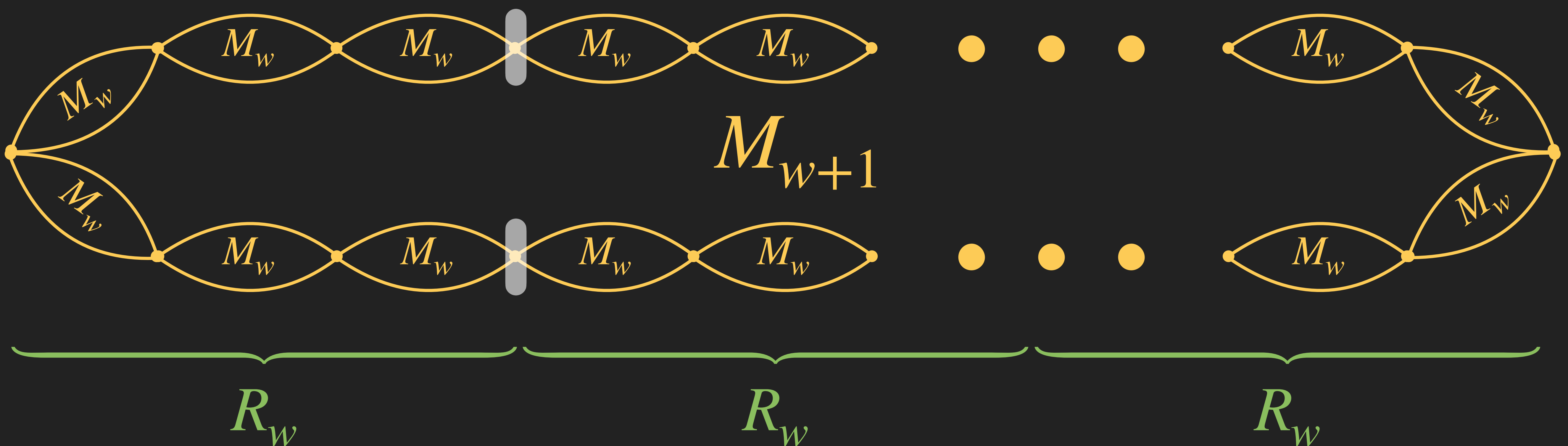
cost $\geq R_w \cdot R_w \cdot \mathrm{diam}(M_w)$

- Construct metric space 

recursively
- Goal: (random) sequence 

s. t. opt $= \mathrm{diam}(M_w)$

cost $\geq R_w \cdot \mathrm{diam}(M_w)$



cost $\geq R_w \cdot R_w \cdot \mathrm{diam}(M_w)$

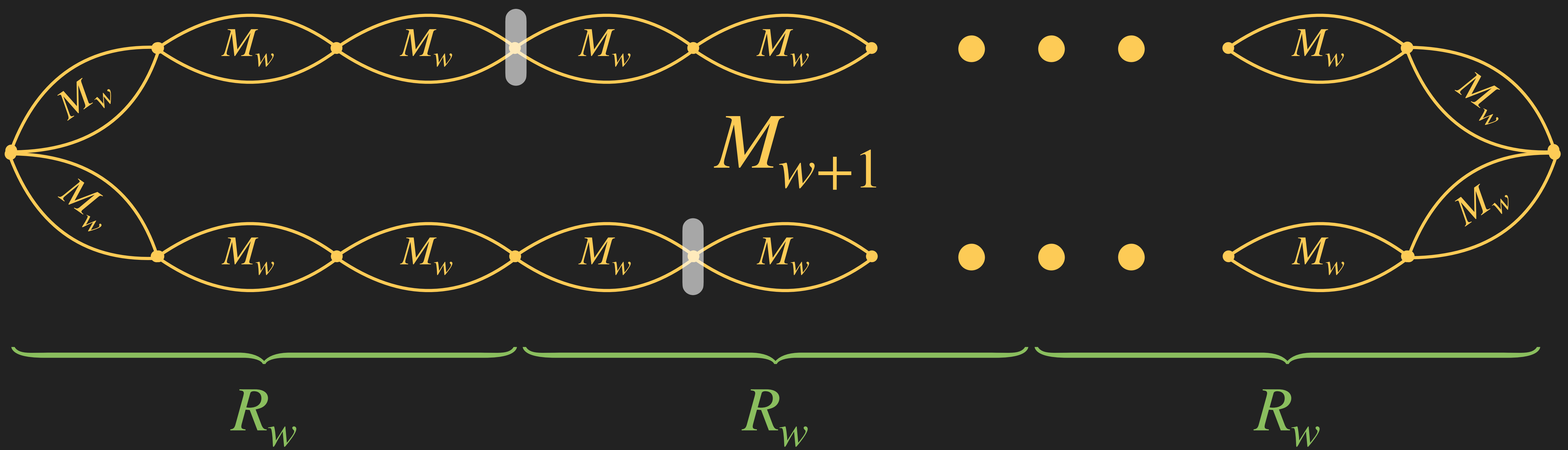
- Construct metric space 

recursively

- Goal: (random) sequence 

s. t. opt $= \mathrm{diam}(M_w)$

  cost $\geq R_w \cdot \mathrm{diam}(M_w)$



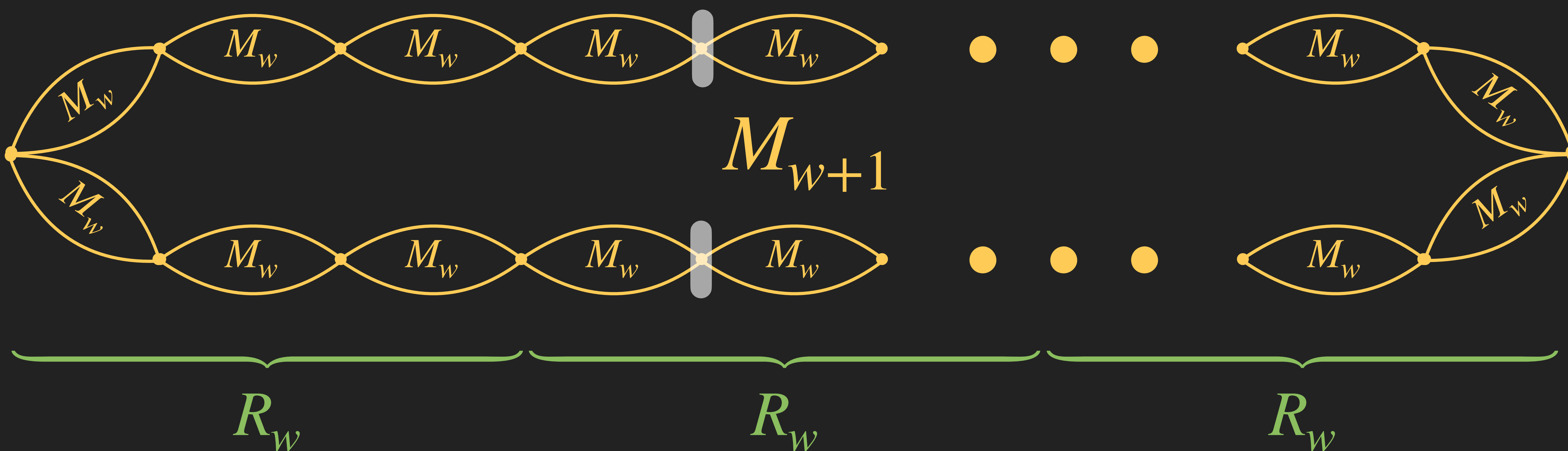
cost $\geq R_w \cdot R_w \cdot \mathrm{diam}(M_w)$

- Construct metric space 

recursively
- Goal: (random) sequence 

s. t. opt $= \mathrm{diam}(M_w)$

cost $\geq R_w \cdot \mathrm{diam}(M_w)$



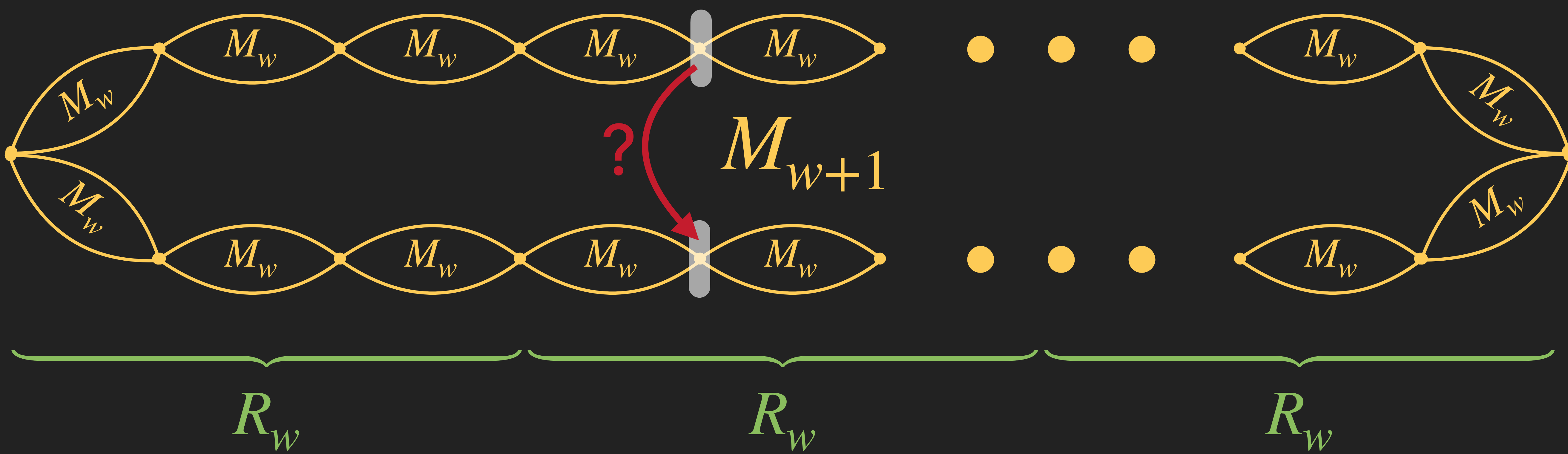
cost $\geq R_w \cdot R_w \cdot \mathrm{diam}(M_w)$

- Construct metric space 

recursively
- Goal: (random) sequence 

s. t. opt $= \mathrm{diam}(M_w)$

cost $\geq R_w \cdot \mathrm{diam}(M_w)$



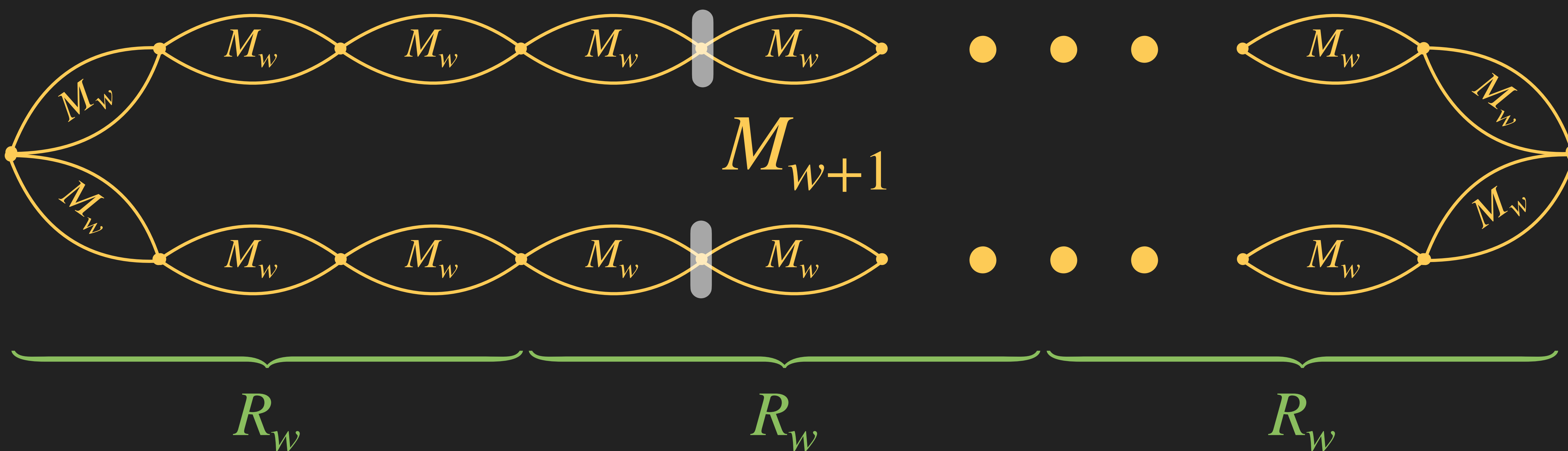
cost $\geq R_w \cdot R_w \cdot \mathrm{diam}(M_w)$

- Construct metric space 

recursively
- Goal: (random) sequence 

s. t. opt $=$ diam(M_w)

cost $\geq R_w \cdot$ diam(M_w)

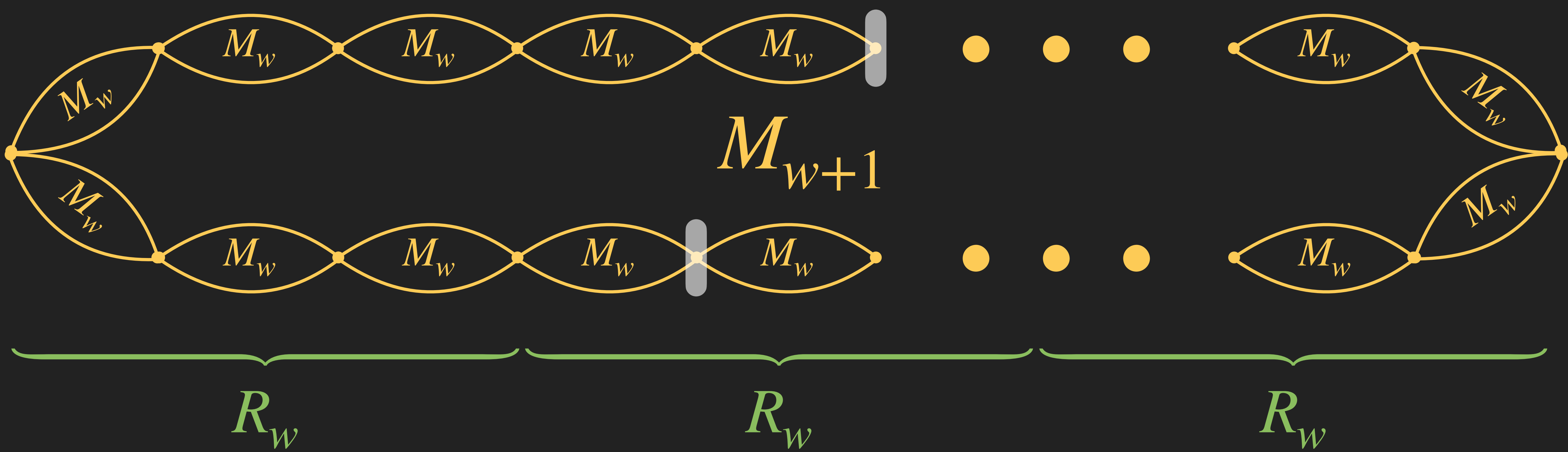


cost $\geq R_w \cdot R_w \cdot$ diam(M_w)

- Construct metric space  recursively

- Goal: (random) sequence  s. t. opt $= \mathrm{diam}(M_w)$

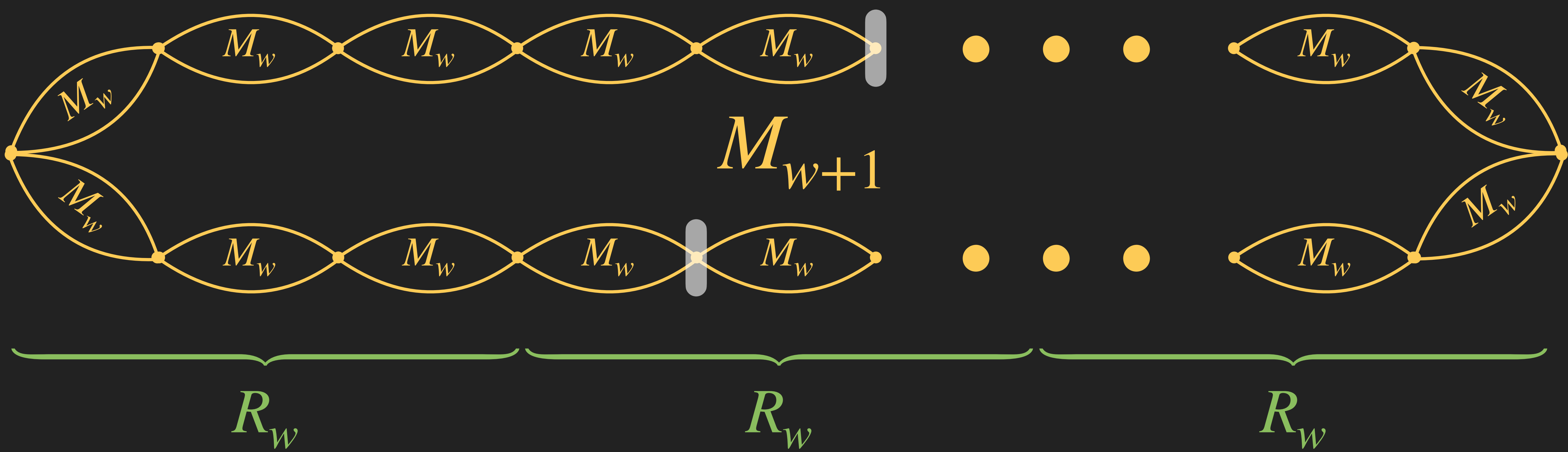
cost $\geq R_w \cdot \mathrm{diam}(M_w)$



cost $\geq R_w \cdot R_w \cdot \mathrm{diam}(M_w)$

- Construct metric space M_w recursively
- Goal: (random) sequence M_w s. t. opt $=$ diam(M_w)

  cost $\geq R_w \cdot$ diam(M_w)



cost $\geq R_w \cdot R_w \cdot$ diam(M_w)

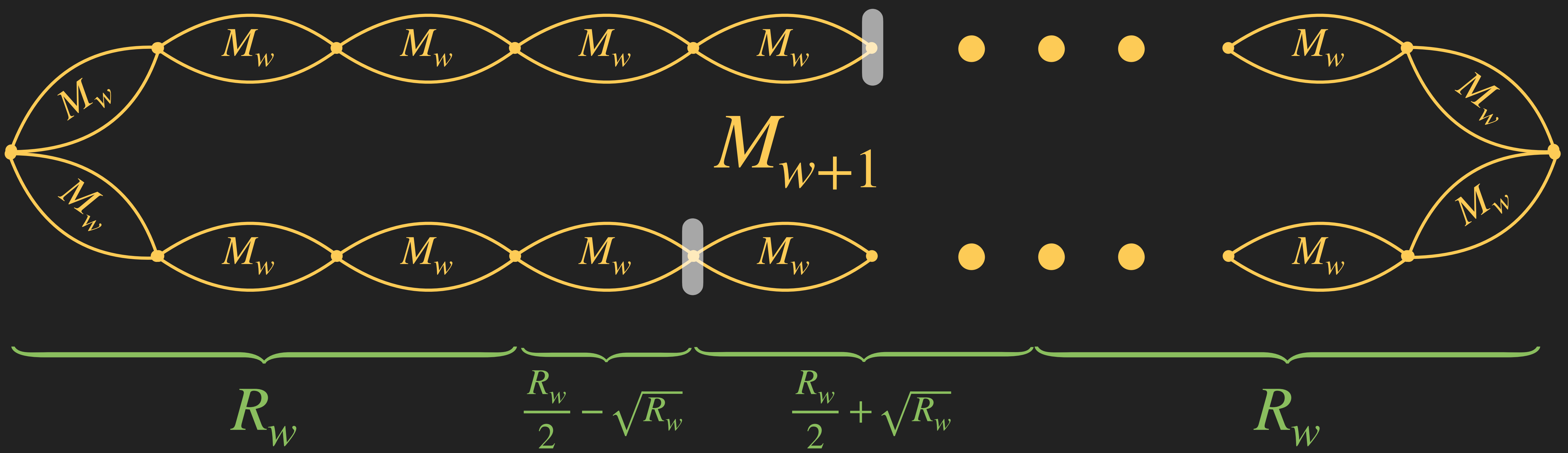
$+ 0.5 \cdot R_w \cdot R_w \cdot$ diam(M_w)

- Construct metric space 

recursively
- Goal: (random) sequence 

s. t. opt $= \mathrm{diam}(M_w)$

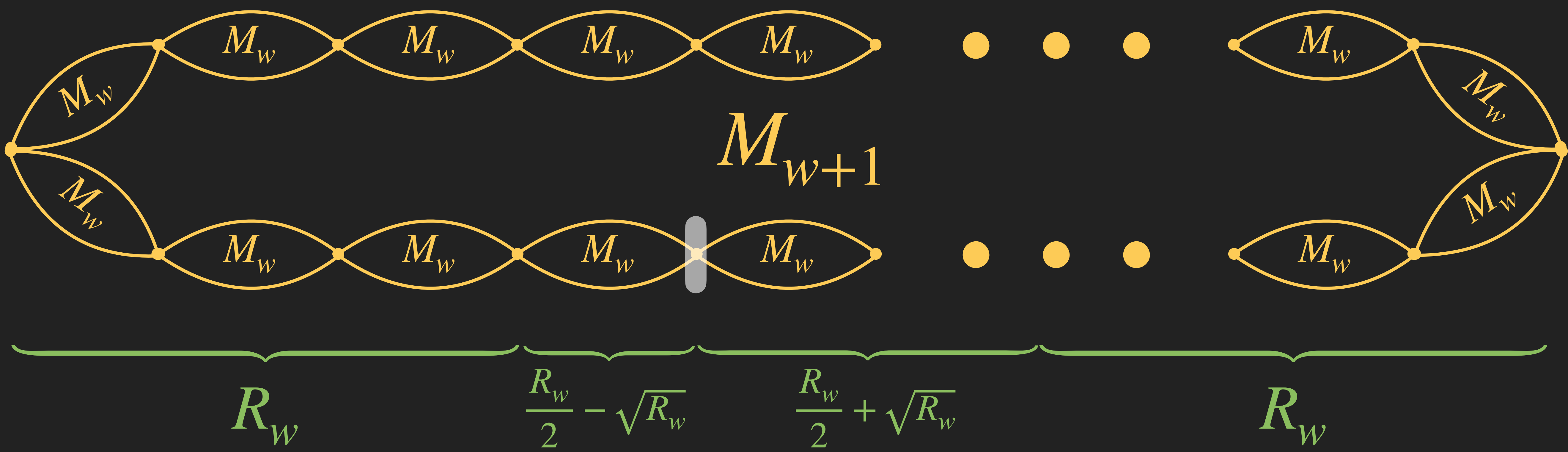
cost $\geq R_w \cdot \mathrm{diam}(M_w)$



$$\text{cost} \geq R_w \cdot R_w \cdot \mathrm{diam}(M_w)$$
$$+ 0.5 \cdot R_w \cdot R_w \cdot \mathrm{diam}(M_w)$$

- Construct metric space M_w recursively
- Goal: (random) sequence M_w s. t. opt $= \mathrm{diam}(M_w)$

  cost $\geq R_w \cdot \mathrm{diam}(M_w)$

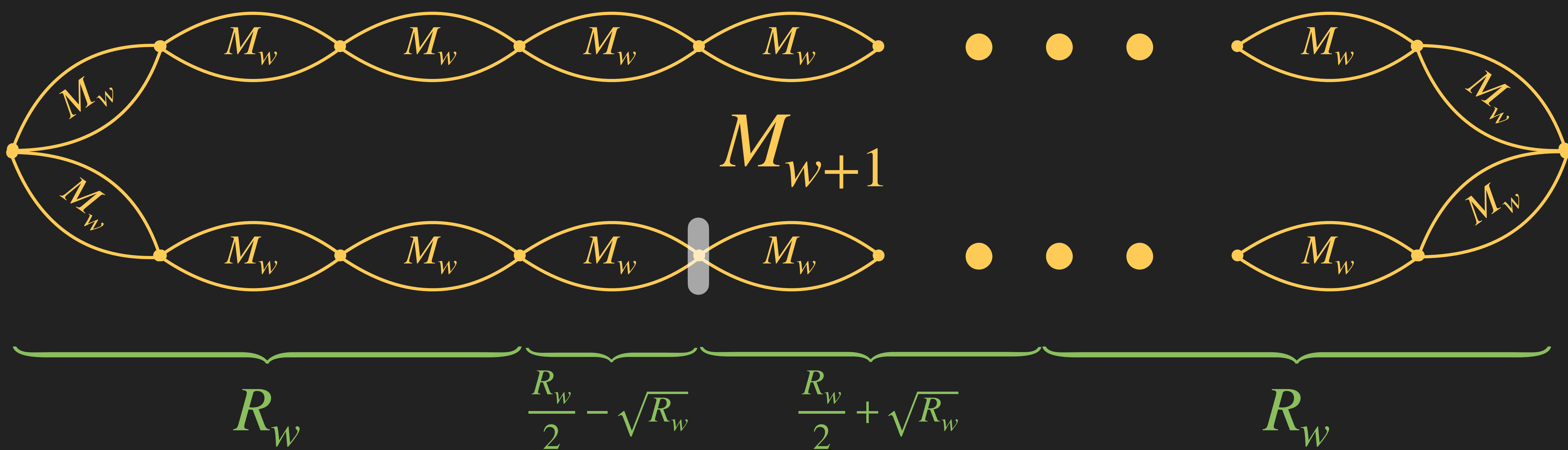


$$\text{cost} \geq R_w \cdot R_w \cdot \mathrm{diam}(M_w) + 0.5 \cdot R_w \cdot R_w \cdot \mathrm{diam}(M_w)$$

- Construct metric space  recursively

- Goal: (random) sequence  s. t. opt $=$ diam(M_w)

cost $\geq R_w \cdot$ diam(M_w)



cost $\geq R_w \cdot R_w \cdot$ diam(M_w)

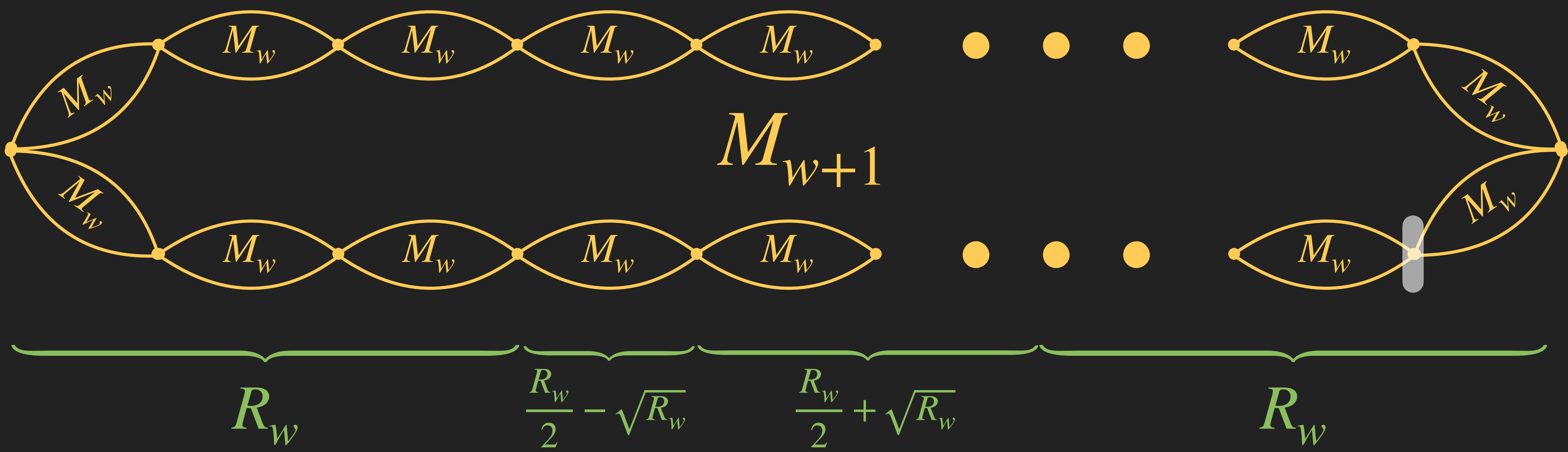
$+\, 0.5 \cdot R_w \cdot R_w \cdot$ diam(M_w)

$+\, (1.5 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot$ diam(M_w)

- Construct metric space  recursively

- Goal: (random) sequence  s. t. opt $=$ diam(M_w)

cost $\geq R_w \cdot$ diam(M_w)



cost $\geq R_w \cdot R_w \cdot$ diam(M_w)

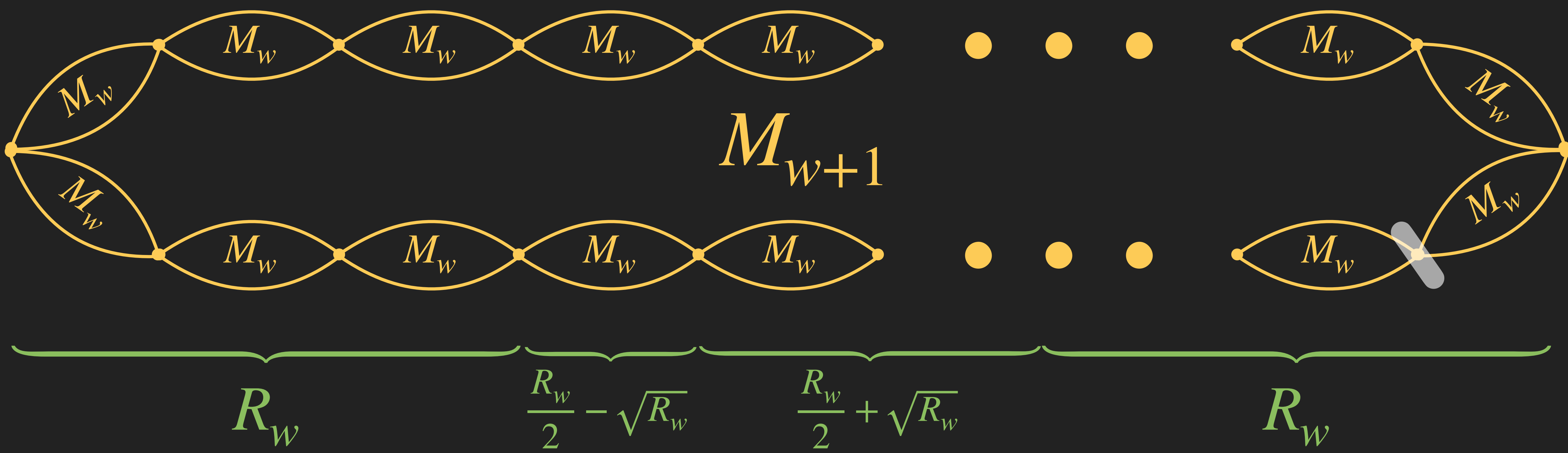
$+\, 0.5 \cdot R_w \cdot R_w \cdot$ diam(M_w)

$+\, (1.5 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot$ diam(M_w)

- Construct metric space  recursively

- Goal: (random) sequence  s. t. opt = diam(M_w)

cost $\geq R_w \cdot$ diam(M_w)



cost $\geq R_w \cdot R_w \cdot$ diam(M_w)

$+ \, 0.5 \cdot R_w \cdot R_w \cdot$ diam(M_w)

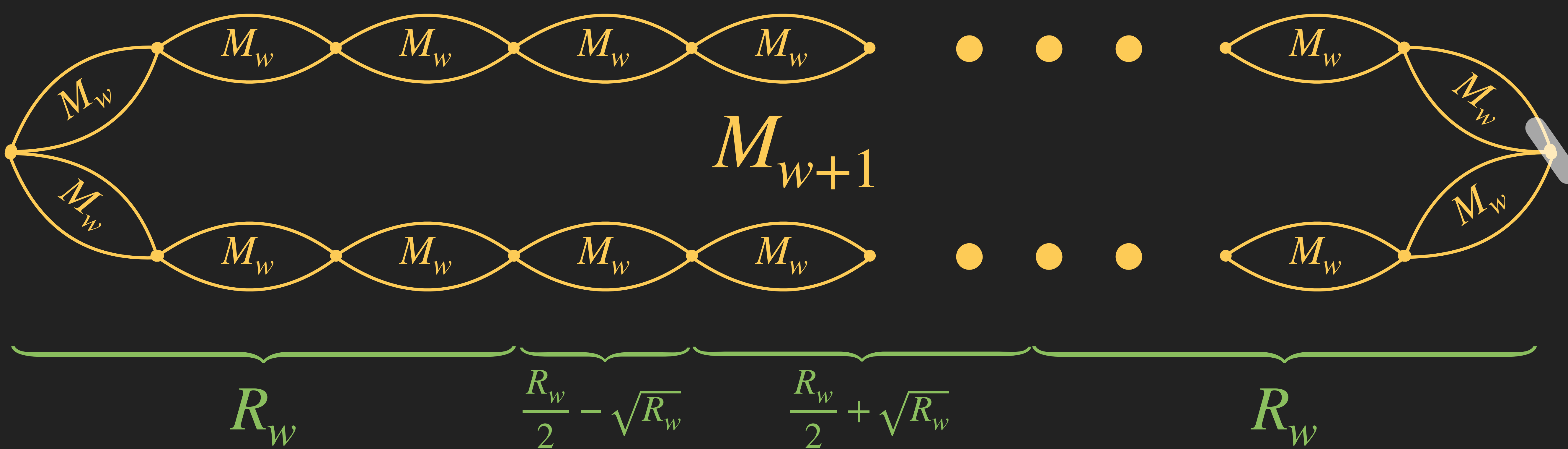
$+ \, (1.5 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot$ diam(M_w)

- Construct metric space 

recursively
- Goal: (random) sequence 

s. t. opt $= \mathrm{diam}(M_w)$

  cost $\geq R_w \cdot \mathrm{diam}(M_w)$

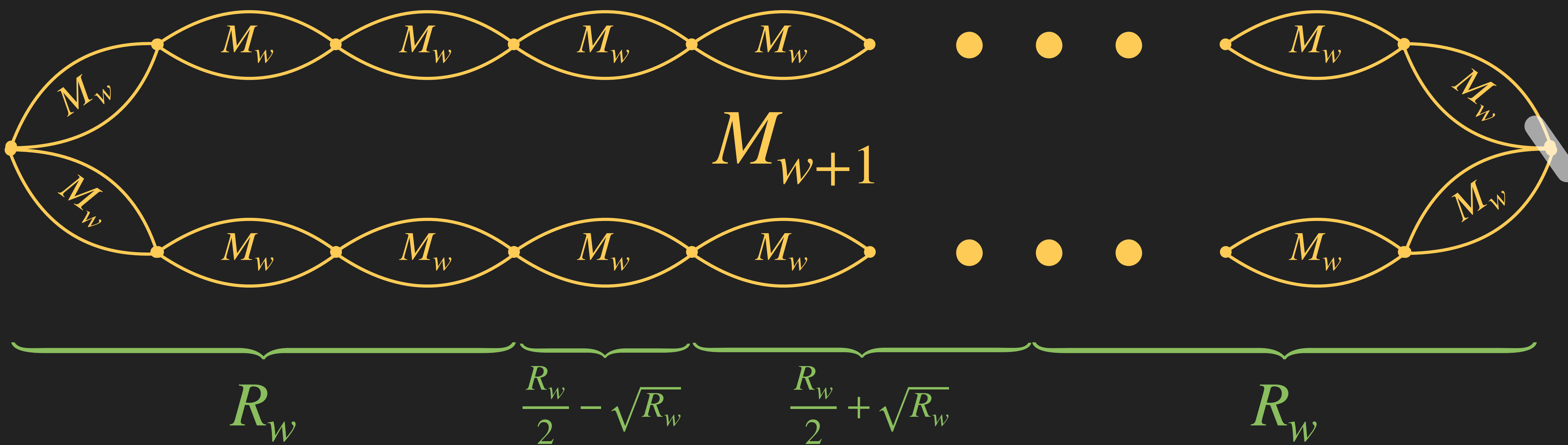


$$\begin{aligned} \text{cost} \geq\ & R_w \cdot R_w \cdot \mathrm{diam}(M_w) \\ & + 0.5 \cdot R_w \cdot R_w \cdot \mathrm{diam}(M_w) \\ & + (1.5 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot \mathrm{diam}(M_w) \end{aligned}$$

- Construct metric space  recursively

- Goal: (random) sequence  s. t. opt $=$ diam(M_w)

  cost $\geq R_w \cdot$ diam(M_w)



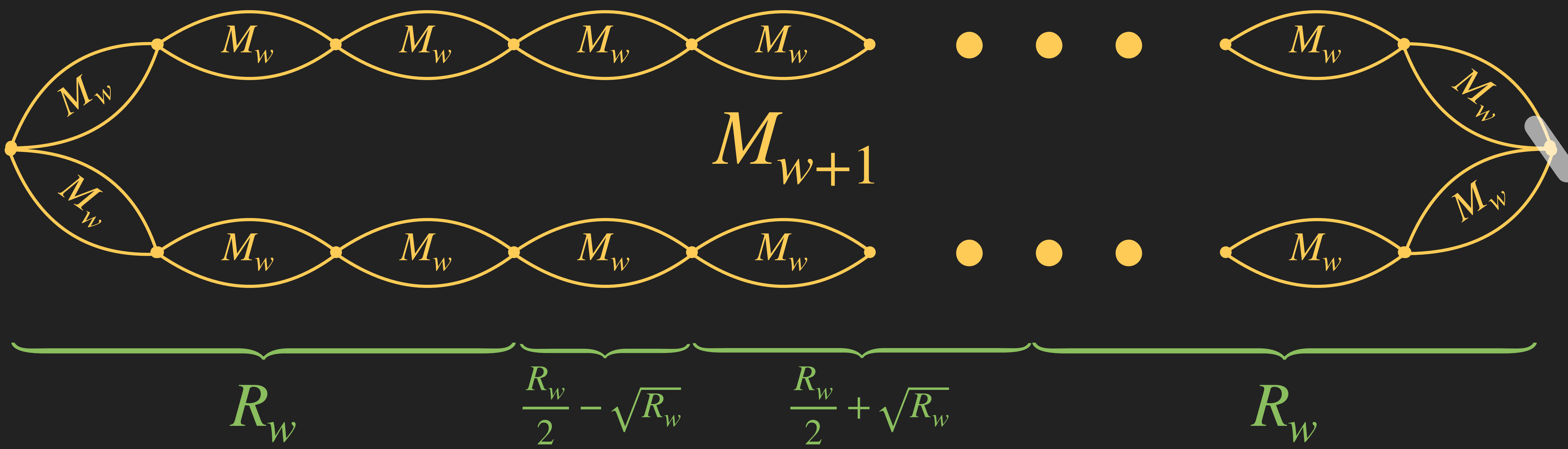
cost $\geq R_w \cdot R_w \cdot$ diam(M_w)

$+ 0.5 \cdot R_w \cdot R_w \cdot$ diam(M_w)

$+ (1.5 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot$ diam(M_w)

- Construct metric space M_w recursively
- Goal: (random) sequence M_w s. t. opt $= \text{diam}(M_w)$

  cost $\geq R_w \cdot \text{diam}(M_w)$



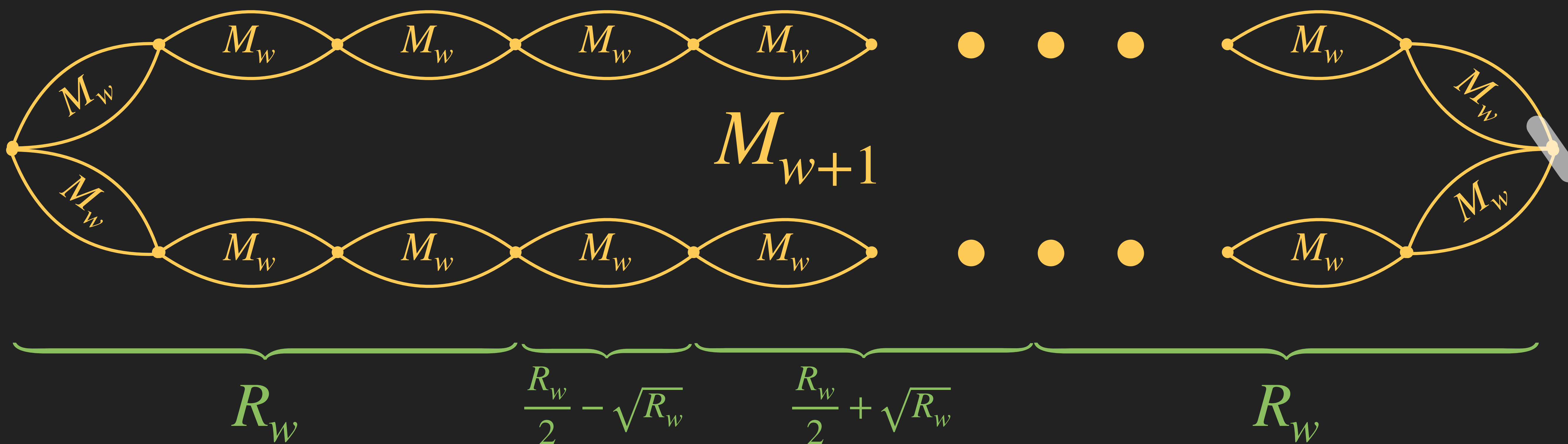
$$
\begin{aligned}
\text{cost} \geq\ & R_w \cdot R_w \cdot \text{diam}(M_w) \\
& + 0.5 \cdot R_w \cdot R_w \cdot \text{diam}(M_w) \\
& + (1.5 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot \text{diam}(M_w) \\
= & \ (3 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot \text{diam}(M_w)
\end{aligned}
$$

- Construct metric space 

recursively
- Goal: (random) sequence 

s. t. opt $=$ diam(M_w)

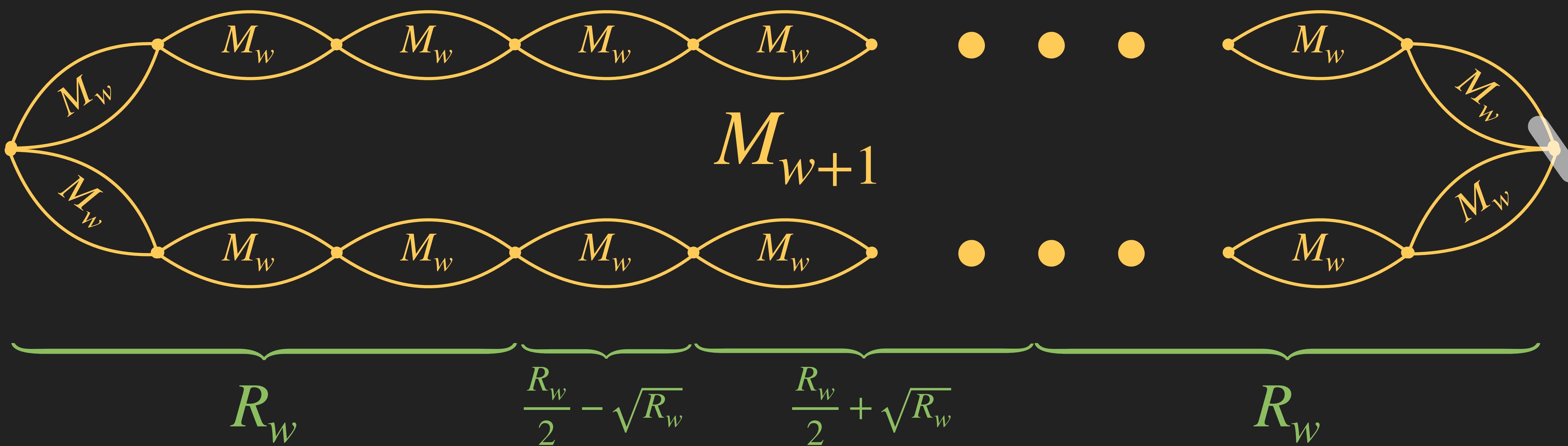
  cost $\geq R_w \cdot$ diam(M_w)



cost $\geq (3 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot$ diam(M_w)

- Construct metric space  recursively
- Goal: (random) sequence  s. t. opt $= \mathrm{diam}(M_w)$

  cost $\geq R_w \cdot \mathrm{diam}(M_w)$



cost $\geq (3 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot \mathrm{diam}(M_w)$

opt $= 3 \cdot R_w \cdot \mathrm{diam}(M_w)$

$$\text{cost} \geq (3 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot \text{diam}(M_w)$$

$$\text{opt} = 3 \cdot R_w \cdot \text{diam}(M_w)$$

$$\text{cost} \geq (3 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot \text{diam}(M_w)$$

$$\text{opt} = 3 \cdot R_w \cdot \text{diam}(M_w)$$

Need $R_{w+1} \leq \dfrac{\text{cost}}{\text{opt}}$, so $R_{w+1} \leq R_w + \dfrac{\sqrt{R_w}}{3}$

$\text{cost} \geq (3 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot \text{diam}(M_w)$

$\text{opt} = 3 \cdot R_w \cdot \text{diam}(M_w)$

Need $R_{w+1} \leq \dfrac{\text{cost}}{\text{opt}}$, so $R_{w+1} \leq R_w + \dfrac{\sqrt{R_w}}{3}$

E.g. $R_w = \dfrac{w^2}{81} = \Omega(w^2)$

$\text{cost} \geq (3 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot \text{diam}(M_w)$

$\text{opt} = 3 \cdot R_w \cdot \text{diam}(M_w)$

Need $R_{w+1} \leq \dfrac{\text{cost}}{\text{opt}}$, so $R_{w+1} \leq R_w + \dfrac{\sqrt{R_w}}{3}$

E.g. $R_w = \dfrac{w^2}{81} = \Omega(w^2)$

$$n = |M_{w+1}| \leq 6R_w |M_w| \leq \prod_{i=1}^{w} 6R_i \leq \prod_{i=1}^{w} i^2 = (w!)^2$$

$$= 2^{O(w \log w)}$$

$$\text{cost} \geq (3 \cdot R_w + \sqrt{R_w}) \cdot R_w \cdot \text{diam}(M_w)$$

$$\text{opt} = 3 \cdot R_w \cdot \text{diam}(M_w)$$

Need $R_{w+1} \leq \dfrac{\text{cost}}{\text{opt}}$, so $R_{w+1} \leq R_w + \dfrac{\sqrt{R_w}}{3}$

E.g. $R_w = \dfrac{w^2}{81} = \Omega(w^2)$

$$n = |M_{w+1}| \leq 6R_w |M_w| \leq \prod_{i=1}^{w} 6R_i \leq \prod_{i=1}^{w} i^2 = (w!)^2$$

$$= 2^{O(w \log w)}$$

$$\implies R_w = \Omega(w^2) = \Omega\left(\left(\frac{\log n}{\log \log n}\right)^2\right)$$

Removing $\log \log$

- Try same with smaller n: use only 6 copies of 

  - $n = |M_w| \leq 6^w$

Removing log log

- Try same with smaller n: use only 6 copies of 

  - $n = |M_w| \leq 6^w$
- Problem:
  - want to flip many coins, but only 3 copies per branch

Removing log log

- Try same with smaller n: use only 6 copies of 

  - $n = |M_w| \leq 6^w$
- Problem:
  - want to flip many coins, but only 3 copies per branch
- Idea:
  - issue recursive request sequence "chunk by chunk"
  - need refined inductive hypothesis

Key Lemma: $\exists$rand. sequence of chunks $\rho_1\rho_2\ldots\rho_m$ and rand. variables $c_1, \ldots, c_m$ s.t.:

- $\mathbb{E}\left[\mathrm{cost}(\rho_i) \mid \rho_1\ldots\rho_{i-1}\right] \geq c_i$
- $\mathbb{E}\left[\sum c_i\right] = \Omega(w^2) \cdot \mathrm{opt}$
- $c_i \approx \mathrm{opt} = \mathrm{diam}(M_w)$

Key Lemma: $\exists$rand. sequence of chunks $\rho_1\rho_2\ldots\rho_m$ and rand. variables $c_1, \ldots, c_m$ s.t.:

- $\mathbb{E}\left[\mathrm{cost}(\rho_i) \mid \rho_1\ldots\rho_{i-1}\right] \geq c_i$
- $\mathbb{E}\left[\sum c_i\right] = \Omega(w^2) \cdot \mathrm{opt}$
- $c_i \approx \mathrm{opt} = \mathrm{diam}(M_w)$

Proof idea:

- biased coins s.t. "cost at top − cost at bottom" is martingale
- martingale CLT/Berry-Esseen yields gap $\pm w \cdot \mathrm{opt}$
- combine small chunks s.t. $c_i \approx \mathrm{opt}$

Implications for other Problems

- Improved LBs for k-taxi, distributed paging, metric allocation
- Similar construction $\Longrightarrow \Omega(k^2)$ for layered graph traversal

Conclusion

- Competitive ratio of MTS is
  - $\Theta(\log n)$ on easiest metrics
  - $\Theta(\log^2 n)$ on hardest metrics
- Competitive ratio of k-server is
  - $\Theta(\log k)$ on easiest metrics with $\geq k+1$ points
  - $\Theta(\log^2 k)$ on hardest metrics with $= k+1$ points
  - $\Omega(\log^2 k) \cap O\left(\min\{\log^2 k \log n, \log^3 k \log \Delta, k\}\right)$ on hardest metrics

Conclusion

- Competitive ratio of MTS is
  - $\Theta(\log n)$ on easiest metrics
  - $\Theta(\log^2 n)$ on hardest metrics
- Competitive ratio of k-server is
  - $\Theta(\log k)$ on easiest metrics with $\geq k+1$ points
  - $\Theta(\log^2 k)$ on hardest metrics with $= k+1$ points
  - $\Omega(\log^2 k) \cap O\left(\min\{\log^2 k \log n, \log^3 k \log \Delta, k\}\right)$ on hardest metrics
- Take-aways: diamond graphs are cool, consider recursion chunk by chunk, look for proof ideas in old poems